



嵌入式系统

Embedded System

张武明

杭州 • 浙江大学 • 2021



第九讲上 串行总线与通信技术

- § 9.1 串行通信的基本概念
- § 9.2 **MCS-51** 串行通信接口
- § 9.3 **MCS-51** 串行口的应用



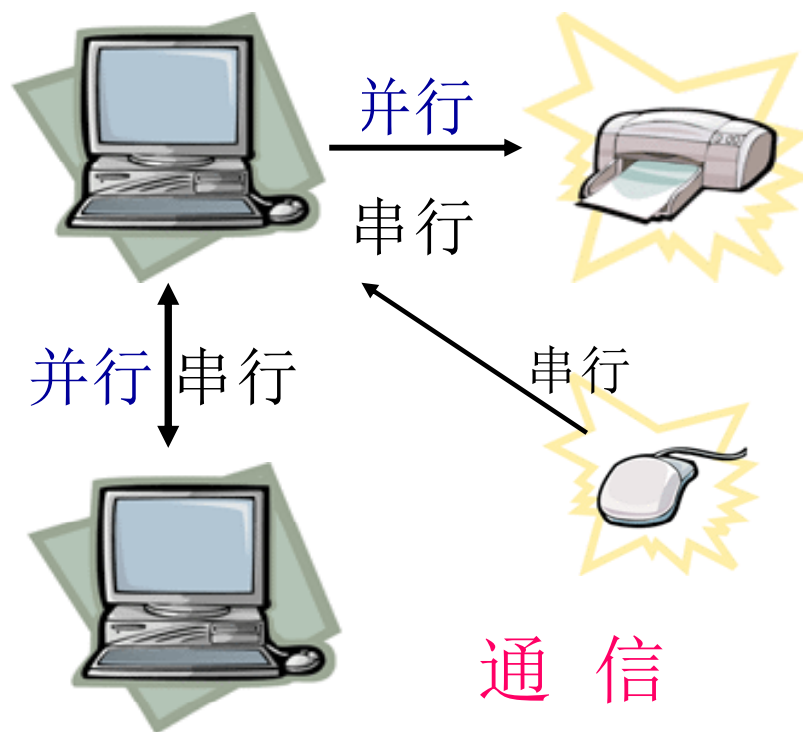
§ 9.1 串行通信的基本概念

- § 9.1.1 通信的基本方式
- § 9.1.2 串行通信的数据传送方式
- § 9.1.3 串行通信的基本类型

§ 9.1.1 通信的基本方式

1. 通信的定义

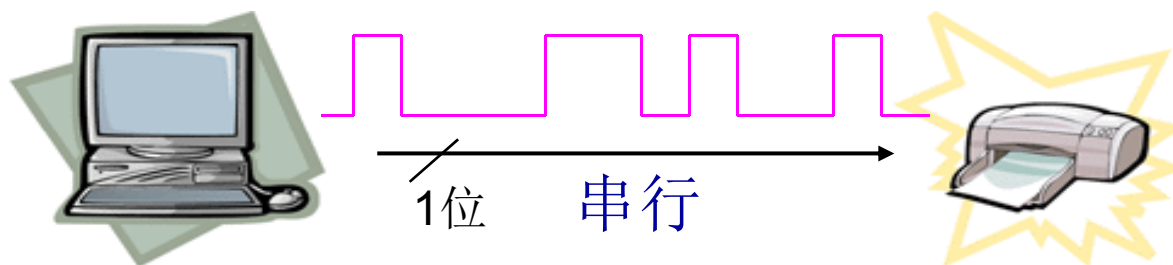
计算机与计算机之间、计算机与设备（I/O设备）之间的信息交换统称为通信。



§ 9.1.1 通信的基本方式

2. 通信的方法

- (1) 并行通信：数据字节的各位同时被传送的通信方式；
- (2) 串行通信：数据字节的各位按顺序逐位传送的通信方式。



- ◆ 传输线少，连线简单；
- ◆ 适用于远距离（可节省大量线路成本）。



§ 9.1.2 串行通信的数据传送方式

按照串行通信过程中信号传输的双方能否同时传送信息分为：

- 1.单工传送方式
- 2.半双工传送方式
- 3.全双工传送方式

§ 9.1.2 串行通信的数据传送方式

● 1. 单工传送方式

单通道工作方式即一种数据只沿一个方向传送的通讯方式。通信信道只有一条，甲只作发送设备，乙只作接收设备。

广播



(乙)



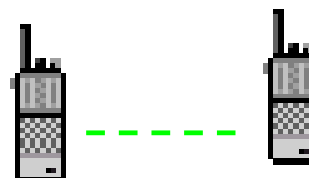
(甲)

§ 9.1.2 串行通信的数据传送方式

- 2.半双工传送方式

数据能在两个方向上传送，但在同一时刻，只能有一个设备发送数据，另一个设备接收数据。

对讲机



通信信道只有一条，数据能在两个方向上传送，但不能同时进行。

§ 9.1.2 串行通信的数据传送方式

● 3.全双工传送方式

两台设备在同一时刻既能接收数据又能发送数据的通信方式。通信双方都具有相互独立的数据发送通道和数据接收通道。

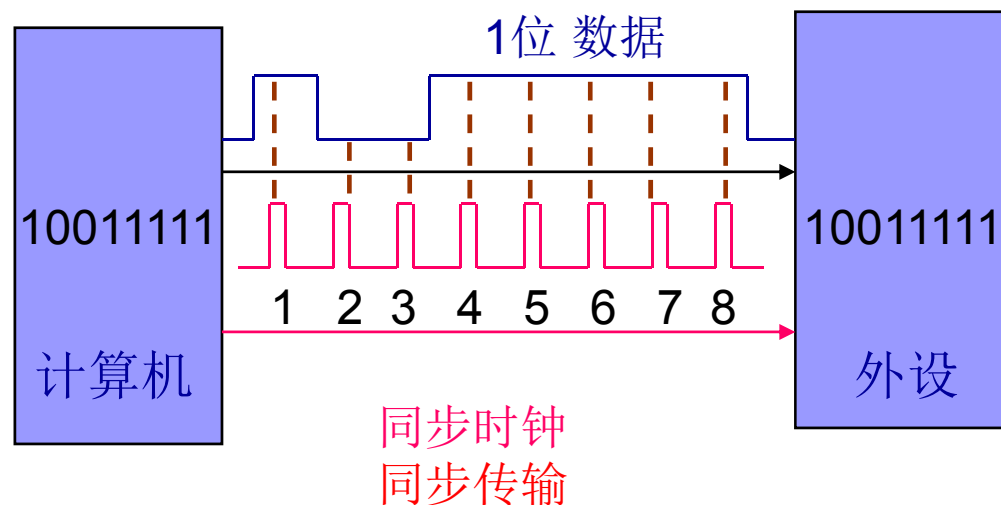
电话(二四线转换)



§ 9.1.3 串行通信的基本类型

- 同步通信方式

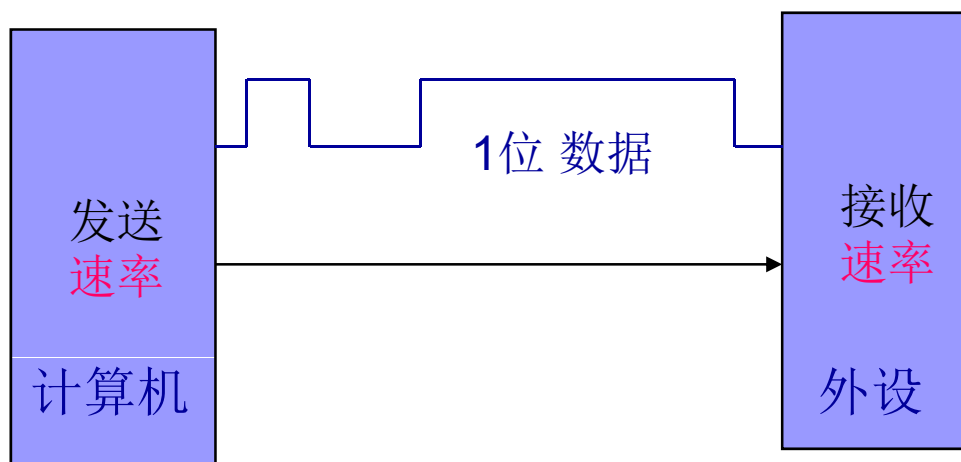
以一串字符为一个传送单位，字符间不加标识位，在一串字符开始用同步字符标识，硬件要求高，通信双方须严格同步。



§ 9.1.3 串行通信的基本类型

- 异步通信方式

以字符为传送单位，用起始位和停止位标识每个字符的开始和结束字符间隔不固定，只需字符传送时同步。

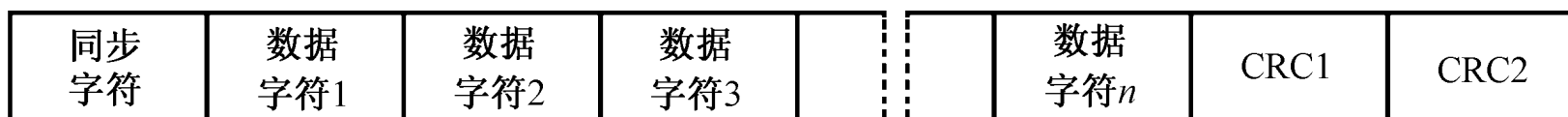


靠发送和接收速率相同来保证

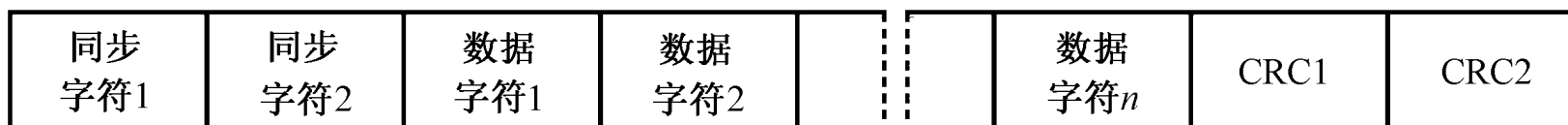
§ 9.1.3 串行通信的基本类型

■ 1. 同步通信格式

数据以“块”为单位，一个数据块包括同步字符、数据及校验字符CRC。



(a) 单同步字符帧结构



(b) 双同步字符帧结构



§ 9.1.3 串行通信的基本类型

通信连线通常采用三线制：

SDA（信号线） **SCL**（时钟线） **GND**（地线）

优点：数据传输速率较高，常用作串行系统总线（内总线），如I²C、SPI、USB等；

缺点：硬件上要求发送时钟和接收时钟保持严格同步。



■ SPI总线

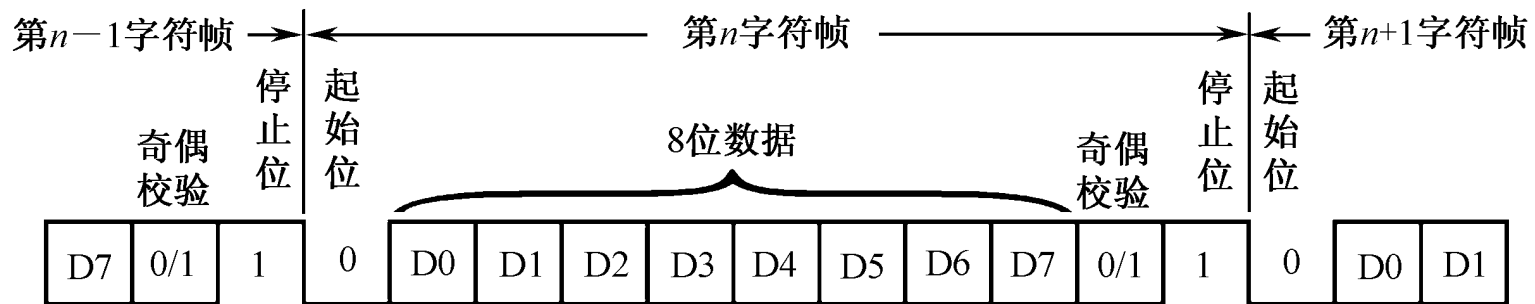
- SPI(Serial Peripheral Interface: 串行外设接口)
- 由三条信号线组成: 串行时钟(SCLK)、数据输出(SDO)、数据输入(SDI)。当有多个从设备时, 增加从设备选择线(从机片选)。
- 用通用IO口模拟SPI总线: 要实现主从设备, 则需输入输出口; 若只实现主设备, 则需输出口即可; 若只实现从设备, 则只需输入口即可。

■ I²C总线

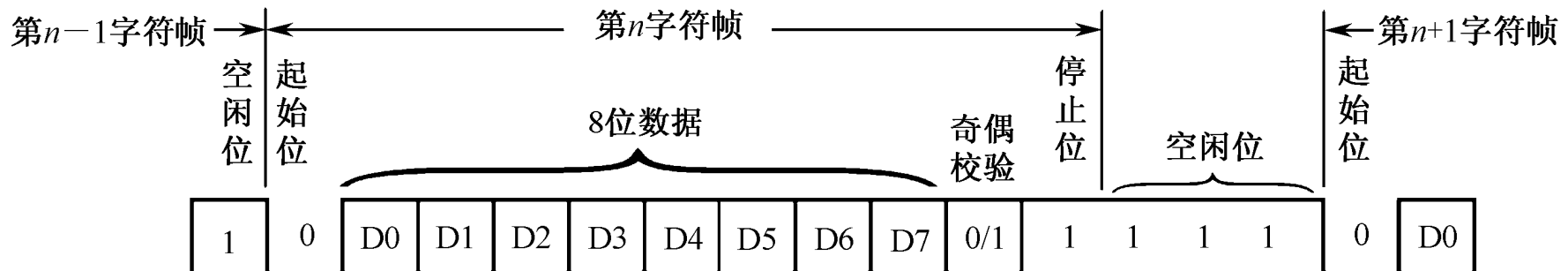
- I²C(INTER IC BUS: 意为IC之间总线)
- 双向、两线(SCL、SDA)、多主控(multi-master)接口标准。传输数据时都会带上目的设备地址, 因此可以实现设备组网
- 用通用IO口模拟I²C总线: 实现双向传输, 需一个输入输出口(SDA), 另外还需一个输出口(SCL)。

§ 9.1.3 串行通信的基本类型

■ 2.异步通信格式



(a) 无空闲位字符帧



(b) 有空闲位字符帧



§ 9.1.3 串行通信的基本类型

- (1) 没有数据发送时，数据线保持“1”状态；
- (2) 发送数据时，先发起始位“0”，其后是数据位，异步传送规定低位在前，高位在后；
- (3) 奇偶位紧跟在数据最高位之后，占用一位（可省去）；
- (4) 数据发送完后再发一位停止位“1”，表示一帧数据结束。同时为接收下一帧数据做好准备。



§ 9.1.3 串行通信的基本类型

通信协议：

- (1) 发送、接收双方的通信速率必须一致；
- (2) 通信双方的数据帧格式必须一致。

优点： 不需要传送同步脉冲，可靠性高，所需设备简单适合远距离通信，常用作串行通信总线（外总线），如RS232、RS485等；

缺点： 数据帧中包含有起始位和停止位以实现同步，从而降低了有效数据的传输速率。



§ 9.1.3 串行通信的基本类型

■ 波特率与传输速率

- (1) 传输速率：每秒钟传输的字符帧数。
- (2) 波特率：异步通信中，双方的通信速率通常以波特率来约定。串口每秒传输的二进制数码的位数称为波特率。

例如数据传送的速率是120字符/秒，而每个字符为10位，则传送波特率为：1200bps（位/秒）



§ 9.1.3 串行通信的基本类型

■ 通信错误的校验

为保证传输过程数据的正确性，在数据传送过程中检测错误的过程称之为校验。最常用的是奇偶校验。

奇偶校验是通过检验被传送的二进制数据中（包括奇偶标志位）“0”或“1”个数的奇偶性，来判断数据传送过程中是否有错。

§ 9.1.3 串行通信的基本类型

■ 例：用奇校验传送十六进制数9EH和35H

解：“10011110” 9EH

“00110101” 35H

为保证传送中每个字符“1”的个数为奇数个，发送的数据帧（每帧11位）如下：

“0 01111001 0 1”

“0 10101100 1 1”

起始位

8位数据位
(低位在前)

校验位

停止位



§ 9.1.4 串行接口的电气特性

■ TTL电平 (LVTTTL3.3V、2.5V)

逻辑1=+2.4~+5V, 接收端 \geq +2.0V

逻辑0= 0~+0.4V, 接收端 \leq +0.8V

■ RS-232C标准 (全双工)

逻辑1=-5~-15V, 接收端-3~-12V

逻辑0=+5~+15V, 接收端+3~+12V

■ RS-485标准 (两线, 半双工)

逻辑1=差分电压 +2~+6V, 接收端 $>$ +0.2V

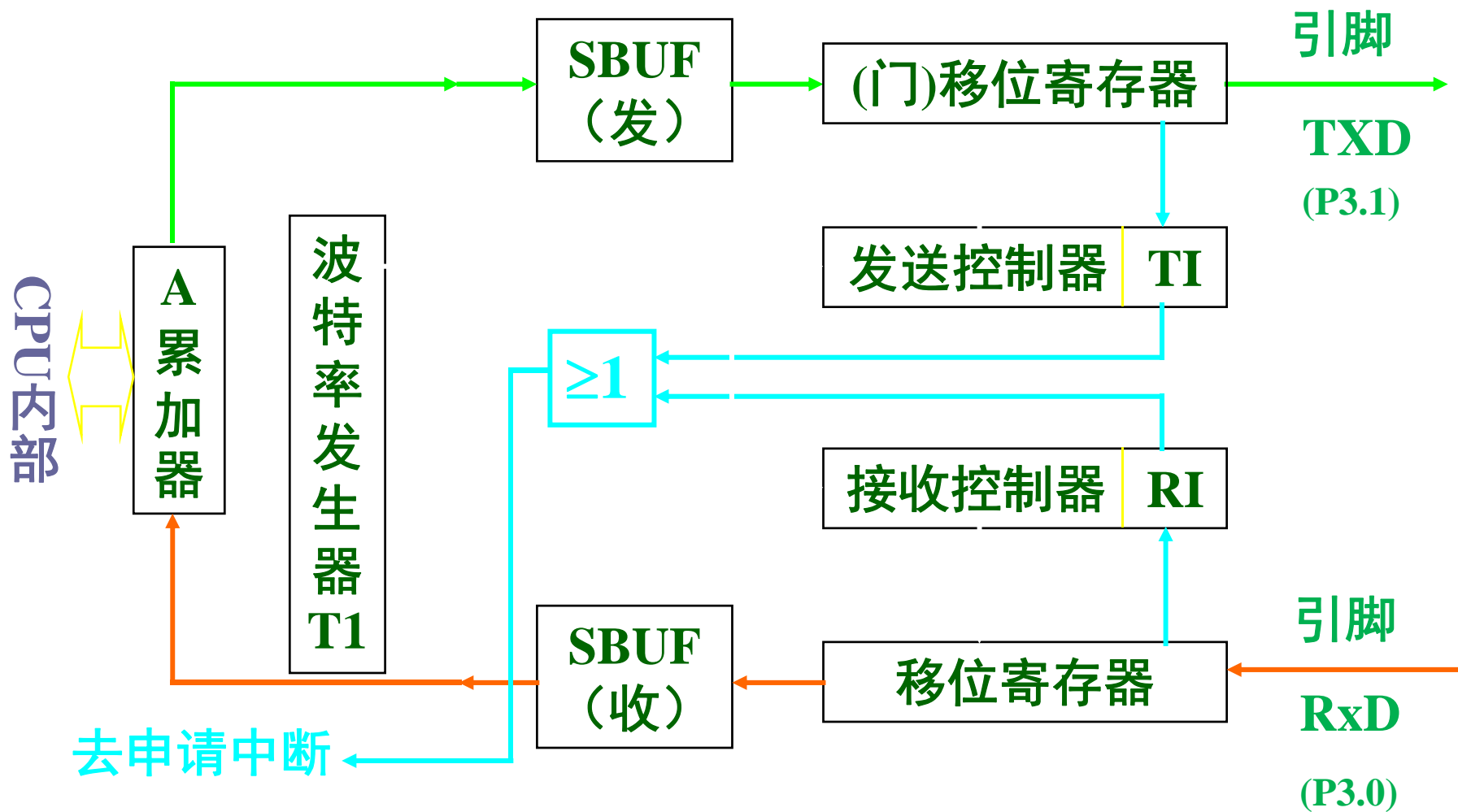
逻辑0=差分电压 -2~-6V, 接收端 $<$ -0.2V



§ 9.2 MCS-51串行通信接口

- § 9.2.1 MCS-51串行口结构
- § 9.2.2 MCS-51串行口的寄存器
- § 9.2.3 MCS-51串行口工作方式

§ 9.2.1 MCS-51 串行口结构






§ 9.2.2 MCS-51串行口的寄存器

1. 串行通信控制寄存器SCON (98H)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
D7	D6	D5	D4	D3	D2	D1	D0

SM0、SM1 控制串行口方式

SM0	SM1	工作方式	功 能 简 述	波特率
0	0	方式0	8位 同步移位寄存器	fosc/12
0	1	方式1	10位 UART	可变
1	0	方式2	11位 UART	fosc/32或/64
1	1	方式3	11位 UART	可变



§ 9.2.2 MCS-51串行口的寄存器

SM2: 允许方式2、3的多机通信特征位。

- 在方式2、3中若SM2=1, 表示接收的第九位数据(RB8)为0时不激活RI。
- 在方式1中若SM2=1, 只有收到有效的停止位时才会激活RI。
- 在方式0中SM2必须为0。

REN: 允许接收控制位。

- REN=1, 允许接收;
- REN=0, 禁止接收。



§ 9.2.2 MCS-51串行口的寄存器

TB8: 发送的第9位数据位，可用作校验位和地址/数据标识位。

RB8: 接收的第9位数据位或停止位。

TI: 发送中断标志，发送一帧结束，**TI=1**，必须软件清零。

RI: 接收中断标志，接收一帧结束，**RI=1**，必须软件清零。

§ 9.2.2 MCS-51串行口的寄存器

2. 电源控制寄存器PCON (87H)

SMOD				GF1	GF0	PD	IDL
D7	D6	D5	D4	D3	D2	D1	D0

SMOD: 波特率选择位。

当SMOD=1, 波特率加倍。

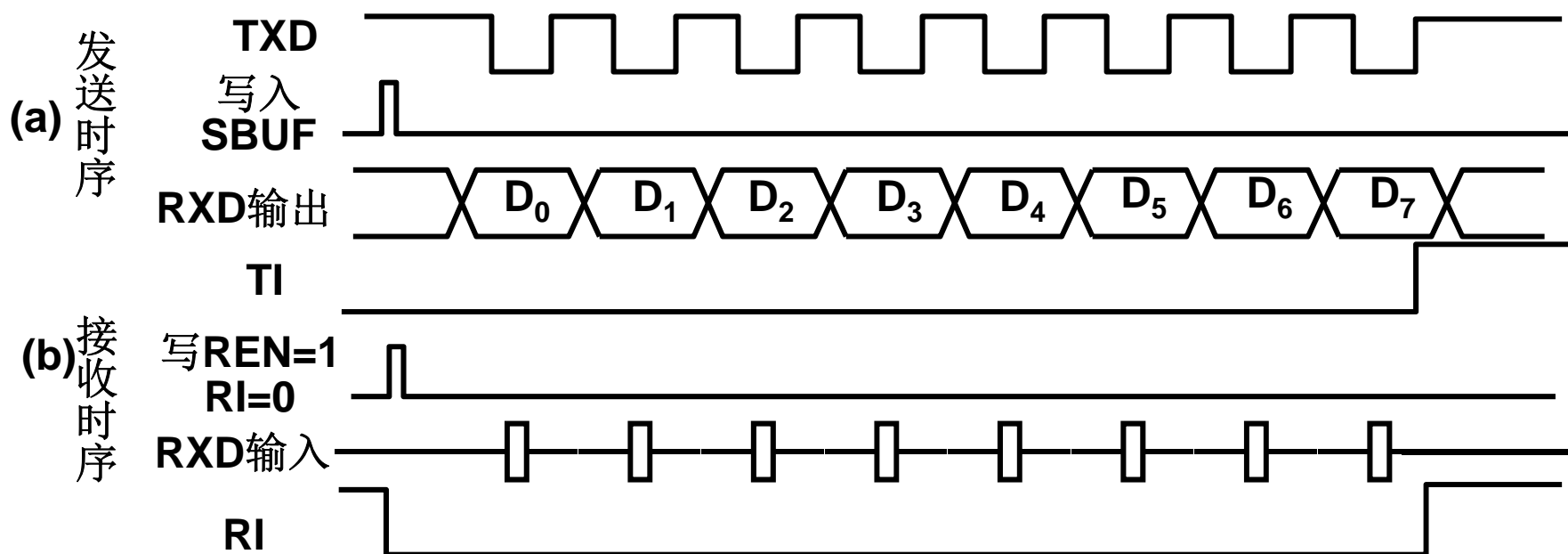
3. 数据缓冲器SBUF (99H)

串行口中有两个在物理意义上相互独立的发送缓冲器和接收缓冲器, 分别用于发送和接收。

§ 9.2.3 MCS-51串行口工作方式

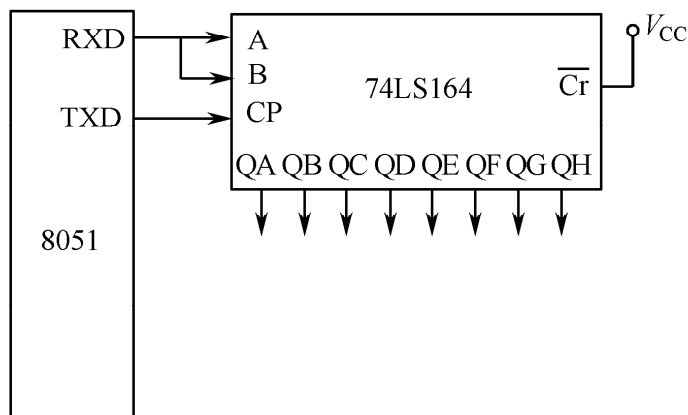
1. 工作方式0（8位移位寄存器I/O方式）

发送/接收过程：SBUF中的串行数据由RXD逐位移出/移入（低位在先，高位在后）；TXD输出移位时钟，频率= $f_{osc}/12$ ；每送出/接收8位数据 TI/RI自动置1；需要用软件清零 TI/RI。

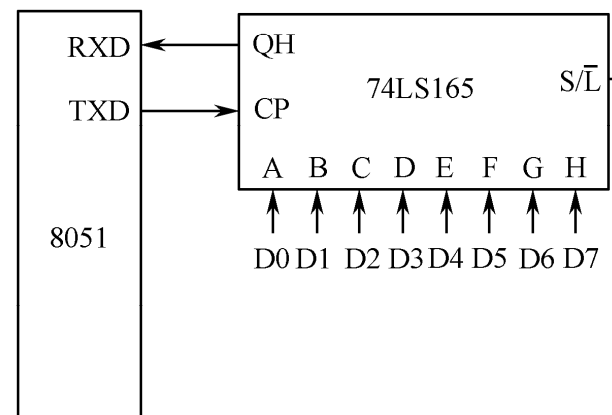


§ 9.2.3 MCS-51 串行口工作方式

注：串行口在方式0下的工作是一种同步通信方式，经常配合“串入并出”“并入串出”移位寄存器一起使用，以达到扩展一个并行口的目的。扩展电路如下图所示。



(a) 扩展输出电路



(b) 扩展输入电路

§ 9.2.3 MCS-51 串行口工作方式

如要发送数据，查询方式的程序如下：

MOV SCON,#00H	； 串行口方式0
MOV SBUF,A	； 将数据送出
JNB TI,\$	； 等待数据发送完毕
CLR TI	； 为下次发送作准备

注意：复位时，SCON 已经被清零。

发送条件：TI=0。

接收条件：RI=0，置位 REN=1（允许接收数据）。



§ 9.2.3 MCS-51串行口工作方式

2. 工作方式1(波特率可变的10位异步通信方式)

发送/接收数据的格式：一帧信息包括1个起始位0，8个数据位和1个停止位1。

发送/接收过程：SBUF中的串行数据由**TXD**/RXD逐位移出/移入；

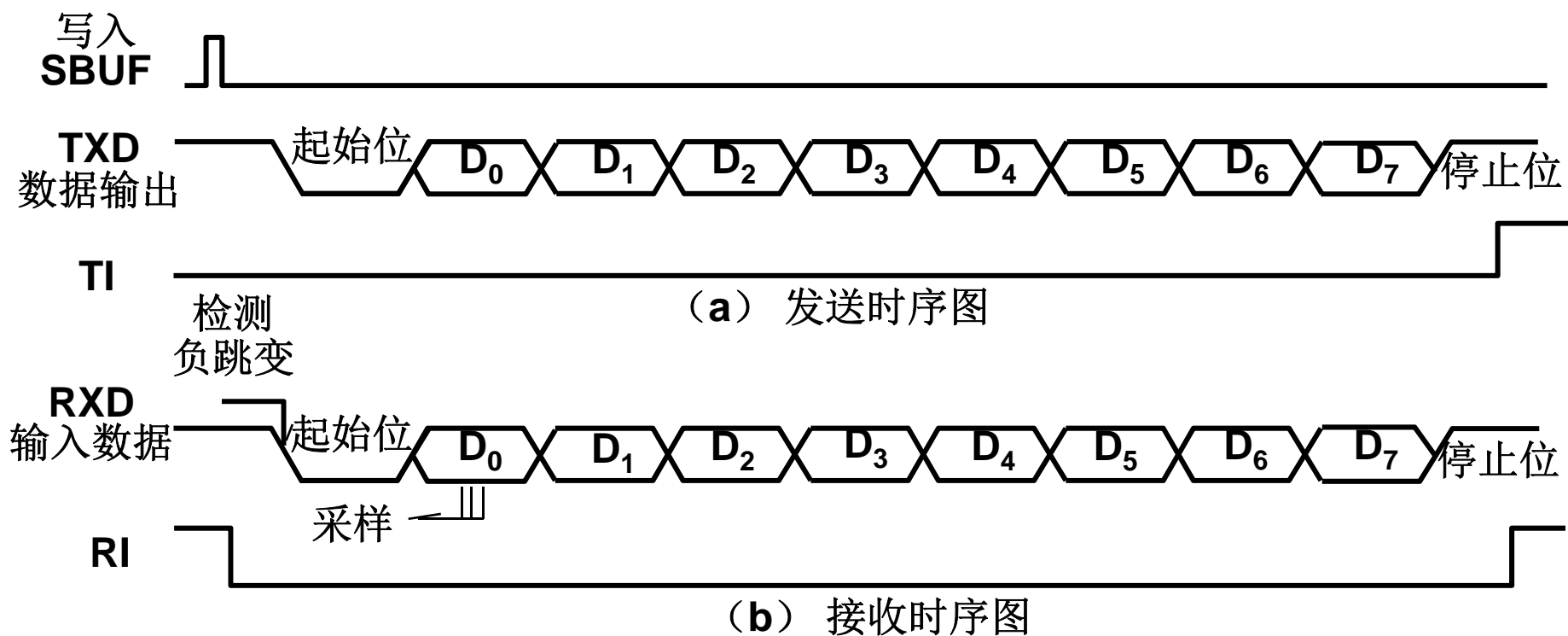
波特率可变： $(2^{\text{SMOD}}/32) \times \text{T1}$ 的溢出率

每送出/接收8位数据 TI/RI自动置1；需要用软件清零 TI/RI。

工作时，发送端自动添加一个起始位和一个停止位；接收端自动去掉一个起始位和一个停止位。

发送/接收条件：同方式0。

§ 9.2.3 MCS-51 串行口工作方式





§ 9.2.3 MCS-51串行口工作方式

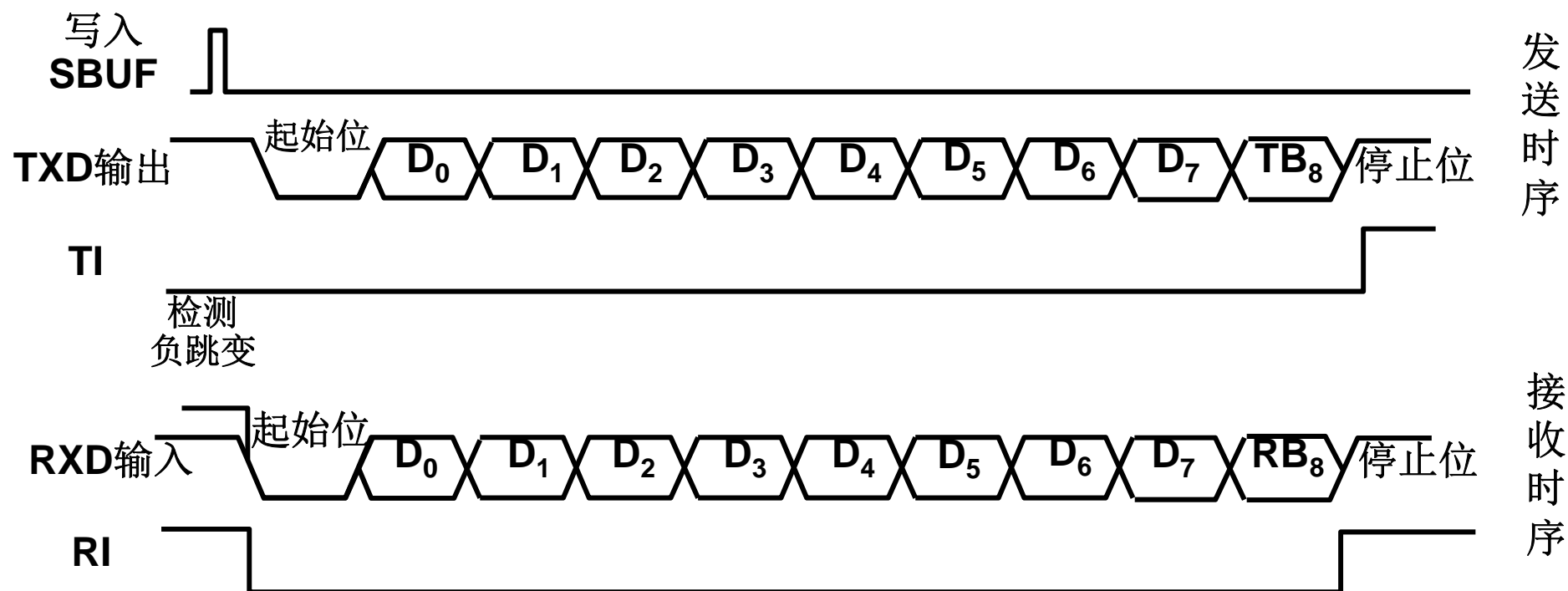
3. 方式2（固定波特率的11位异步接收/发送方式）

发送/接收过程：方式2的接收/发送过程类似于方式1，所不同的是它比方式1增加了一位“第9位”数据(TB8/RB8)，用于“奇偶校验”。方式2常用于单片机间通信。波特率 = $f_{osc} \times 2^{SMOD}/64$ 。

4. 方式3（可变波特率的11位异步接收/发送方式）

方式3和方式2唯一的区别是波特率机制不同，方式3的波特率 = $(2^{SMOD}/32) \times T1$ 的溢出率。

§ 9.2.3 MCS-51串行口工作方式



§ 9.2.4 波特率的设定

- 方式0为固定波特率:

$$B=f_{osc}/12$$

- 方式2可选两种波特率:

$$B=(2^{SMOD}/64) \times f_{osc} \quad (SMOD=0 \text{ 或 } 1)$$

- 方式1和方式3为可变波特率，由定时器溢出速率决定。

$$B=(2^{SMOD}/32) \times T1$$

§ 9.2.4 波特率的设定

- 定时器T1作波特率发生器

定时器T1作波特率发生器时，一般工作在方式2，
即8位系数自动重载方式

$$T = (2^8 - \text{初值}) \times 12 / f_{\text{osc}}$$

$$\text{波特率} = 2^{\text{SMOD}} / 32 \times (\text{T1溢出率}) = 2^{\text{SMOD}} \times f_{\text{osc}} / [32 \times 12 \times (2^n - \text{初值})]$$

$$\text{当 } n=8, \text{ 波特率} = 2^{\text{SMOD}} \times f_{\text{osc}} / [32 \times 12 \times (2^8 - \text{初值})]$$



§ 9.3 MCS-51串行口的应用

- § 9.3.1 串行口的发送和接收
- § 9.3.2 方式0的应用
- § 9.3.3 多机通信与应用

§ 9.3.1 串行口的发送和接收

■ 1、串行口初始化

串行口初始化编程格式：

SIO: MOV SCON, #控制状态字; 写方式字且TI=RI=0

(MOV PCON, #80H) ; 波特率加倍

MOV TMOD, #20H ; T1作波特率发生器

MOV TL1, #X ; 送时间初值

MOV TH1, #X

SETB TR1 ; 启动波特率发生器

§ 9.3.1 串行口的发送和接收

■ 2. 发送程序

(1) 查询方式:

```
TRAM:    MOV  A, @R0          ; 取数据
          MOV  SBUF, A        ; 发送一个字符
WAIT:    JBC  TI, NEXT        ; 等待发送结束
          SJMP WAIT
NEXT:    INC  R0              ; 准备下一次发送
          SJMP TRAM
```

§ 9.3.1 串行口的发送和接收

(2) 中断方式:

ORG	0023H	;	串行口中断入口
	AJMP SINT		
MAIN:	...	;	初始化编程
TRAM:	MOV A, @R0	;	取数据
	MOV SBUF, A	;	发送第一个字符
H:	SJMP H	;	其它工作
SINT:	CLR TI	;	中断服务程序
	INC R0		
	MOV A, @R0	;	取数据
	MOV SBUF, A	;	发送下一个字符
	RETI		


§ 9.3.1 串行口的发送和接收

- 例：将**50—5FH**的**16**个数据由串行口输出，采用方式**3**，偶校验。

波特率为1200bps，用T1作波特率发生器（方式2），重装常数0E8H。在数据写入SBUF前，先将奇偶位写入TB8，即准备好发送的第9位。

$$1.2K = 2^0 \times 11.0592M / [32 \times 12 \times (2^8 - 232)]$$

(P194 常用波特率设置)



§ 9.3.1 串行口的发送和接收

源程序：

```
TR:  MOV  SCON,#0C0H ; 方式3
      MOV  TMOD,#20H  ; T1为方式2
      CLR  TR1
      MOV  TH1,#0E8H
      MOV  TL1,#0E8H
      SETB TR1
      MOV  R0,#50H
      MOV  R7,#10H
```

§ 9.3.1 串行口的发送和接收

```
LOOP1:  MOV  A,@R0
        MOV  C,P                ; 校验位传送到TB8
        MOV  TB8,C
        MOV  SBUF,A            ; 启动串行发送
WAIT:   JNB  TI, WAIT          ; TI必须清零
        CLR  TI
        INC  R0
        DJNZ R7,LOOP1
        RET
```

如 (A) = 33H, 则 P=0; (A) = 13H, 则 P=1。对于偶校验要使发送数据中的“1”的个数为偶数个将 P 送 TB8 即可; 对于奇校验将 P 的反送 TB8 即可。

§ 9.3.1 串行口的发送和接收

■ 3.接收程序

例：与发送子程序对应，同样用方式**3**，偶校验。

源程序：

```
RVE:      MOV  TMOD,#20H   ; T1为方式2
          CLR  TR1
          MOV  TH1,#0E8H
          MOV  TL1,#0E8H
          SETB TR1          ; 启动波特率
          MOV  R0,#50H
          MOV  R7,#10H
          MOV  SCON,#0D0H; 方式3, REN=1
```

§ 9.3.1 串行口的发送和接收

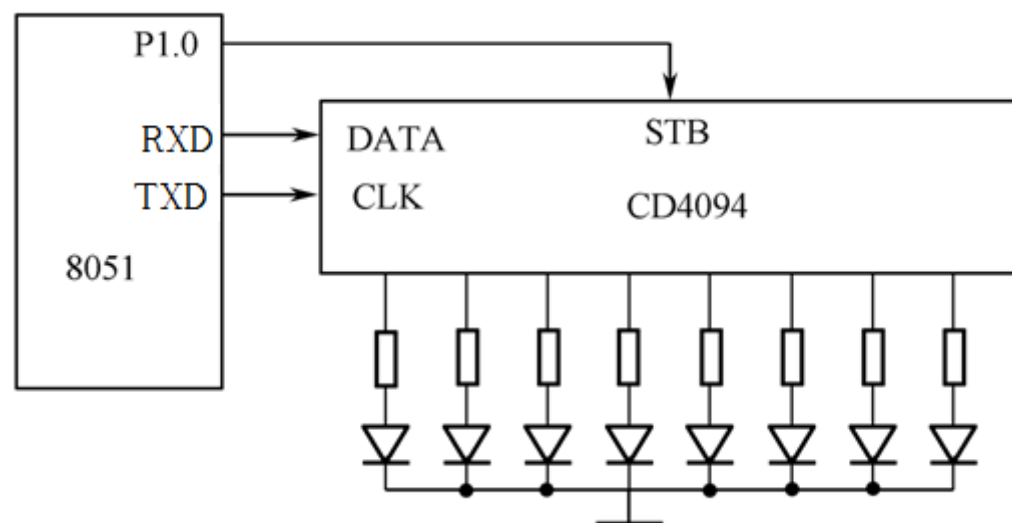
```
W1:      JNB  RI,W1      ; 为1并清零
          CLR  RI
          MOV  A,SBUF    ; 读数据到A, P中已反映奇偶性
          JNB  P, N
          JNB  RB8,E      ; 查第9位信息
          SJMP R
N:        JB   RB8,E
R:        MOV  @R0,A      ; 数据传送正确
          INC  R0
          DJNZ R7,W1
          CLR  F0          ; 置正确标志
          RET
E:        SETB F0          ; 置出错标志
          RET
```

```
if(P!=RB8)
{
    F0=1;
}
else
{
    F0=0;
    .....
}
```

§ 9.3.2 方式0的应用

- 用8051串行口外接CD4094扩展8位并行输出口，8位并行口的各位都接一个发光二极管，要求发光二极管呈流水灯状态（轮流点亮）。

解：硬件连接电路如下图所示。



（移位寄存器74LS164、74HC164）



§ 9.3.2 方式0的应用

开始通信之前，应先对控制寄存器SCON进行初始化。将00H送SCON即设置方式0。数据传送采用查询方式，通过查询TI的状态，来决定是否发送下一帧数据。在串行接收时，通过对RI查询来确定何时接收下一帧数据。程序如下：



§ 9.3.2 方式0的应用

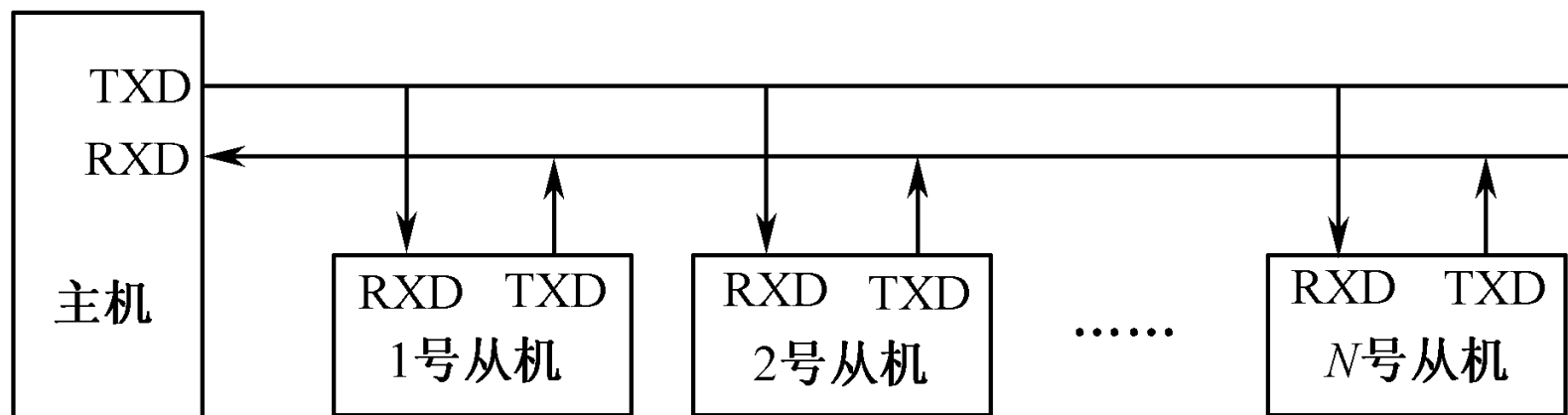
```
ORG 1000H
START: MOV SCON, #00H    ;置串行口工作方式0
      MOV A, #80H        ;最高位灯先亮
      CLR P1.0           ;关闭并行输出
OUT0:  MOV SBUF, A        ;开始串行输出
OUT1:  JNB TI, OUT1       ;输出是否结束
      CLR TI             ;清TI标志
```




§ 9.3.2 方式0的应用

	SETB P1.0	; 打开并行口输出
	ACALL DELAY	; 延时一段时间
	RR A	; 循环右移
	CLR P1.0	; 关闭并行输出
	SJMP OUT0	; 循环
DELAY:	MOV R7, #250	; 延时子程序
D1:	MOV R6, #250	
D2:	DJNZ R6, D2	
	DJNZ R7, D1	
	RET	
	END	

§ 9.3.3 多机通信与应用





§ 9.3.3 多机通信与应用

1.多机通信原理

利用SCON中的SM2及串口工作方式2或3的第9位数据TB8来实现。当串口以方式2或方式3接收时，若SM2=1，那么仅当RB8=1时8位数据才会送入SBUF,接收的第9位数据到RB8，RI置1，否则自动丢失该帧信息，不产生中断。

当SM2=0时，第9位数据状态不影响接收与否。



§ 9.3.3 多机通信与应用

2.通信协议

- 主机置SM2位0，所有从机的SM2位置1，处于接收地址帧状态。
- 主机发送一地址帧，其中，8位是地址，第9位为1表示该帧为地址帧。
- 所有从机收到地址帧后，都将接收的地址与本机的地址比较。对于地址相符的从机，使自己的SM2位置0（以接收主机随后发来的数据帧），并把本站地址发回主机作为应答；对于地址不符的从机，仍保持SM2=1，对主机随后发来的数据帧不予理睬。
- 从机发送数据结束后，要发送一帧校验和，并置第9位（TB8）为1，作为从机数据传送结束的标志。



§ 9.3.3 多机通信与应用

- 主机接收数据时先判断数据接收标志（RB8），若接收帧的RB8=0，则存储数据到缓冲区，并准备接收下帧信息。若RB8=1，表示数据传送结束，并比较此帧校验和，若正确则回送正确信号00H，此信号命令该从机复位（即重新等待地址帧）；若校验和出错，则发送0FFH，命令该从机重发数据。发送数据。
- 主机收到从机应答地址后，确认地址是否相符，如果地址不符，发复位信号（数据帧中TB8=1）；如果地址相符，则清TB8，开始发送数据。
- 从机收到复位命令后回到监听地址状态（SM2=1）。否则开始接收数据和命令。

§ 9.3.3 多机通信与应用

3.多机通信举例---主机（发送数据用中断方式）

例：主机将外部RAM 2000H开始的255个数据串行传送给地址为68H的从机，从机将接收的数据存到外部RAM 1000H开始处。波特率由定时器1产生（方式2），串行口工作在方式3。

```
甲机发送:      ORG    0000H
                  LJMP   MAIN
                  ORG    0023H
                  LJMP   SIOINT
MAIN:            MOV     DPTR,#2000H
                  MOV     TMOD,#20H ;T1工作方式2
                  MOV     TH1,#0F4H
```

§ 9.3.3 多机通信与应用

```
MOV    TL1,#0F4H
MOV    PCON,#00H      ; 波特率为2400
CLR    ET1             ; 关T1中断
CLR    ES              ; 关串行中断
SETB   EA              ; CPU开中断
SETB   TR1             ; 启动T1波特率发生器
MOV    SCON,#0C0H      ; 串行方式3
SETB   TB8              ; TB8=1, 表示发送地址
MOV    SBUF,#68H        ; 先发从机的地址码
W1:    JNB   TI,W1       ; 查询发送是否完成
CLR    TI              ; 清标志
SETB   ES              ; 开串行中断
CLR    TB8              ; TB8=0, 准备发送数据
MOV    R2,#0FFH        ; 置计数器初值
```

§ 9.3.3 多机通信与应用

```
MOVX    A,@DPTR        ; 发送第一个字节
MOV     SBUF,A
SJMP    $               ; 主程序
```

串行中断程序:

```
SIOINT:  CLR     TI                ; 一帧发送完进入中断
          DJNZ    R2,W3            ; 判断255个数据是否发完
          AJMP    W4
W3:      INC     DPTR
          MOVX    A,@DPTR          ; 继续发送
          MOV     SBUF,A
W4:      RETI                     ; 中断返回
```


§ 9.3.3 多机通信与应用

4. 多机通信举例---从机（用中断方式）

乙机接收:

```
ORG    0000H
LJMP   MAIN
ORG    0023H
LJMP   SIOINT
ORG    0100H
MAIN:  MOV    TMOD,#20H
        MOV    TH1,#0F4H
        MOV    TL1,#0F4H
        MOV    PCON,#00H      ; 波特率=2400bps
        SETB   EA
        CLR    ET1
        SETB   ES
```

§ 9.3.3 多机通信与应用

```
                SETB    TR1
                MOV     SCON,#0F0H    ; REN=1方式3接收
                MOV     R2,#0FFH      ; 数据个数
                MOV     DPTR,#1000H    ; 接收数据存放首地址
HERE:           SJMP    HERE

串行口中断:
SIOINT:         MOV     C,RB8
                JNC     Z              ; C=0表示收到的是数据
                MOV     A,SBUF         ; 接收地址信息
                CJNE    A,#68H, N      ; 不是本机地址, 返回
                CLR     SM2            ; 清SM2, 准备接收数据
N:              CLR     RI
                RETI
```



§ 9.3.3 多机通信与应用

```
Z:      MOV      A,SBUF
        MOVX     @DPTR,A    ; 存到DPTR为地址的外部数据RAM
        INC      DPTR
        CLR      RI          ; 清中断标志
        DJNZ     R2,AGAIN    ; 判断是否接收完
        SETB     F0          ; 置接收完标志
AGAIN:  RETI
```



嵌入式系统

Embedded System

张武明

杭州 • 浙江大学 • 2021



第九讲下 人机接口技术

- **§ 9.4** 键盘接口与设计
- **§ 9.5** LED显示接口与设计



§ 9.4 键盘接口与设计

- § 9.4.1 键盘输入基础知识
- § 9.4.2 独立式键盘接口设计
- § 9.4.3 矩阵式键盘接口设计



§ 9.4.1 键盘输入基础知识

- 键盘是单片机控制系统最常用、最简单的输入设备。用户可以通过键盘输入数据或命令，实现简单的人机通信。
- 键盘与单片机的接口包括
 - 硬件：键盘的组织即键盘结构及其与主机的连接方式；
 - 软件：对按键操作的识别与分析，称为键盘管理程序。

键盘管理程序的任务：

- 1) 识键：判断是否有键按下。若有则进行译码；若无，则等待或转别的工作；
- 2) 译键：识别出哪一键被按下并求出被按下键的键值；
- 3) 键值分析：根据键值找出对应处理程序的入口并执行之。



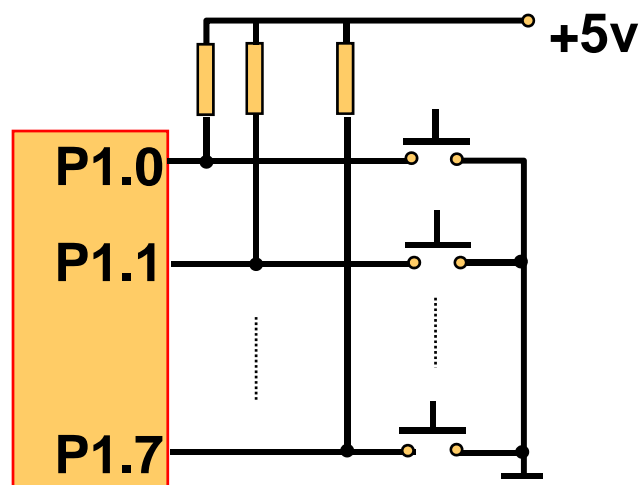
§ 9.4.1 键盘输入基础知识

■ 键盘结构形式:

- (1) **编码式键盘**，除了键开关外，还有专门的硬件电路，用于识别闭合键并产生键代码，但是这种方式硬件结构复杂，成本高，常用的大规模集成电路键盘编码器如 8279，BC7281等；
- (2) **非编码式键盘**，仅由键开关组成，其它工作如键识别、键代码的产生、去抖动等不是由硬件完成而是由软件完成的。为了简化硬件电路，降低成本，目前单片机控制系统中大多数采用非编码键盘。

§ 9.4.1 键盘输入基础知识

■ 非编码键盘： 独立式键盘



每键相互独立，各自与一条I/O线相连，CPU可直接读取该I/O线的高/低电平状态。

优点：硬软件结构简单，判键速度快，使用方便；

缺点：占I/O口线多。

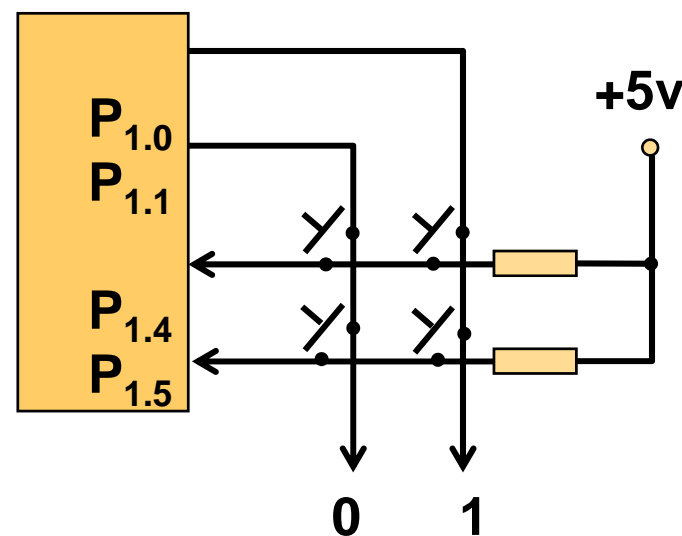
适用场合：多用于设置控制键、功能键。适用于键数较少的场合。

§ 9.4.1 键盘输入基础知识

■ 非编码键盘:

对于矩阵式结构, $m \times n$ 个按键只需要 $m+n$ 条口线; 而独立式结构, m 个按键需要 m 条口线。

矩阵式键盘



键按矩阵排列,各键处于矩阵行/列结点处,CPU通过对行(列)I/O线送已知电平信号,然后读取列(行)线状态信息。逐线扫描,得出键码。

特点: 键多时占用I/O口线少,硬件资源利用合理,但判键速度慢。

适用场合: 多用于设置数字键,适用于键数多的场合。



§ 9.4.1 键盘输入基础知识

■ 键盘的工作方式

CPU对键盘进行扫描时：

- 要及时，以保证能扫描到每一次的按键动作；
- 扫描不能占用CPU过多的时间。

■ 键盘的三种工作方式：

- 编程扫描方式（查询方式）
- 定时扫描方式
- 中断工作方式



§ 9.4.1 键盘输入基础知识

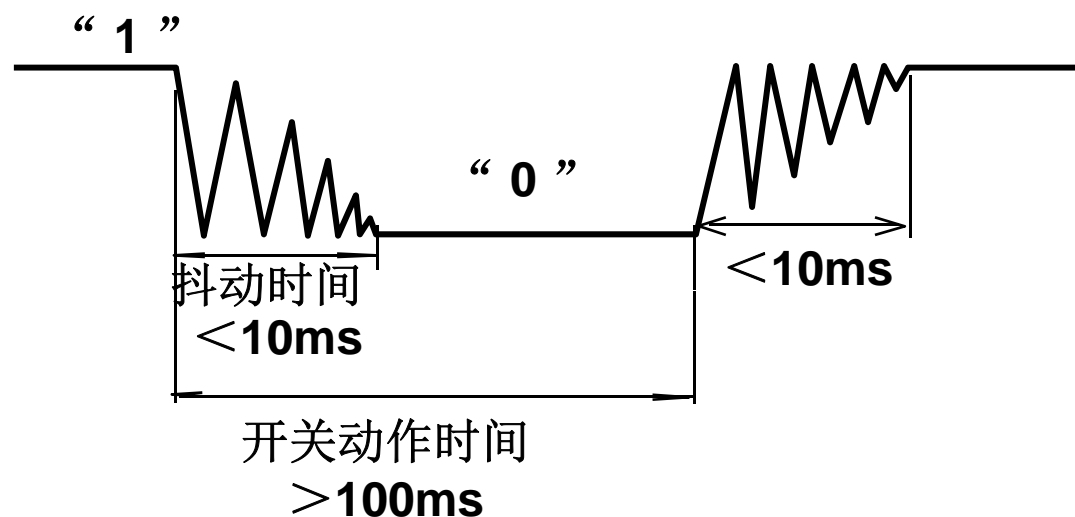
- 1) 查询方式：利用CPU在完成其他工作的空余，调用键盘扫描程序。当CPU在运行其他程序时，它不会再响应按键操作。
- 2) 定时扫描方式：用定时器产生定时中断，CPU响应中断后便对键盘进行扫描，并转入相应的键功能处理程序。定时中断周期一般应小于50ms。
- 3) 中断工作方式：当有键按下时，利用硬件产生外部中断请求，CPU响应中断后对键盘进行扫描，并转入相应的键处理程序。

§ 9.4.1 键盘输入基础知识

■ 键盘的可靠性

1、键抖动及消除

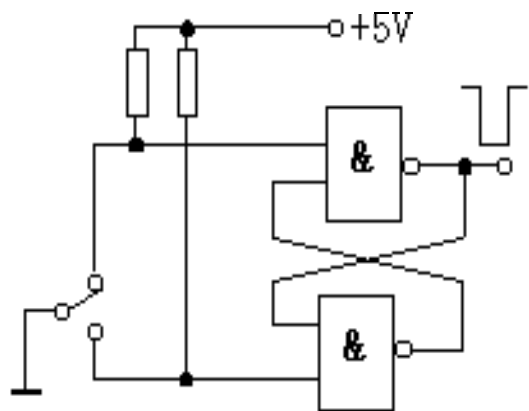
触点式开关按键按下或释放时，由于机械弹性作用，通常伴随有一定时间的触点抖动。抖动过程如图。



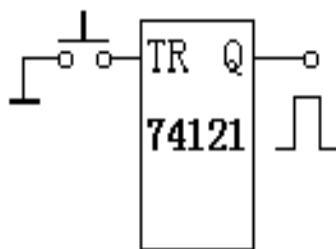
§ 9.4.1 键盘输入基础知识

- 键抖动可能导致计算机对一次按键操作作出多次响应，所以要去抖动。

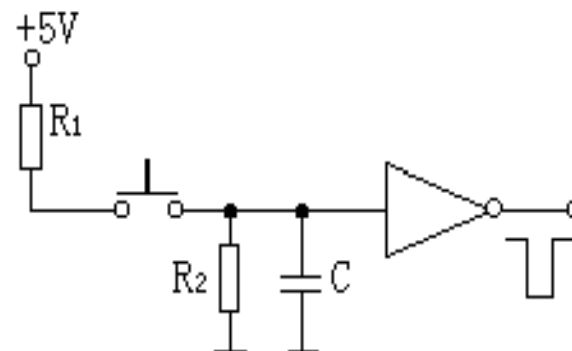
1) 硬件消抖电路：



(a) 双稳态消抖电路



(b) 单稳态消抖电路



(c) 滤波消抖电路

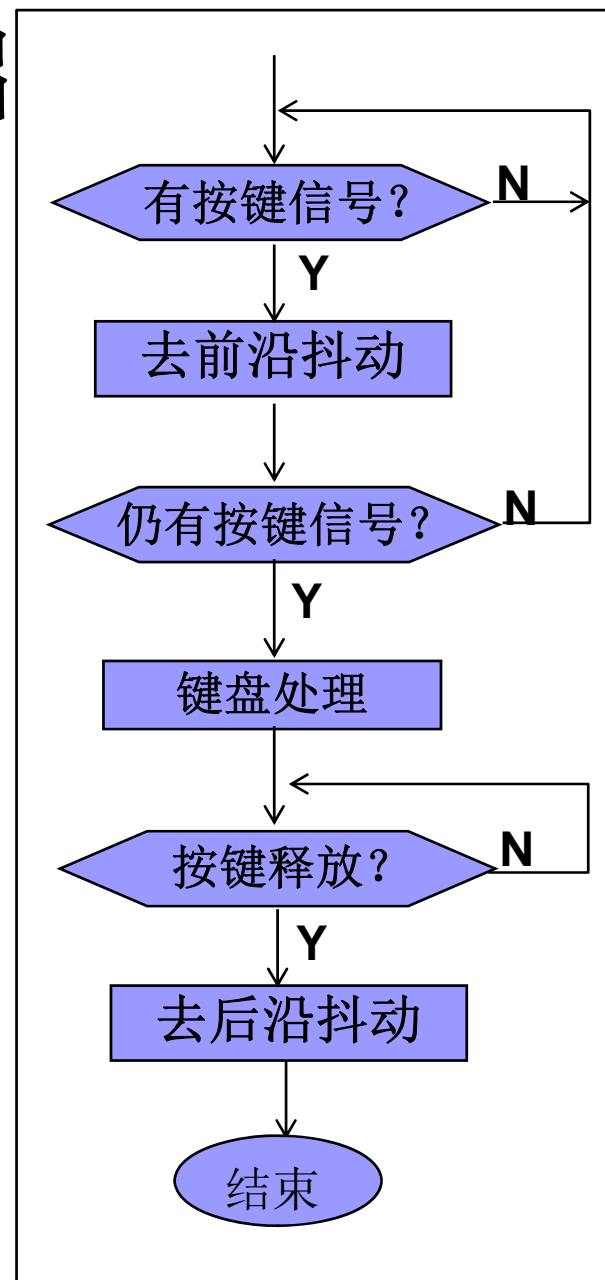
- ### 2) 软件延时法：
- 当检测到有键按下时，用软件延时 10ms~20ms，等待键稳定后重新判断键是否按下。

§ 9.4.1 键盘输入基础

■ 2、键连击的处理

一次按键操作被多次执行的现象称为连击。

连击现象的消除



§ 8.4.1 键盘输入基础

■ 2、键连击的处理

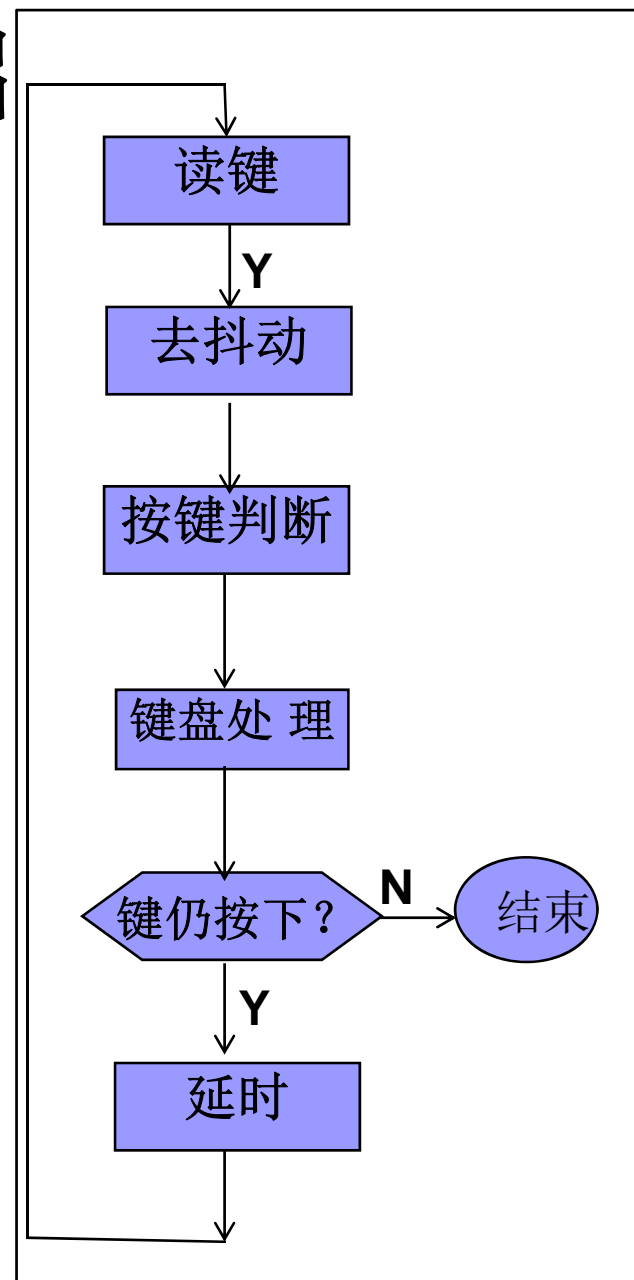
一次按键操作被多次执行的现象称为连击。

连击现象的消除

连击现象的利用：

合理利用连击现象，有时会给设计带来方便。

延时模块是为了控制和调节连击的速度。





§ 9.4.1 键盘输入基础知识

■ 3、串键保护与实现

串键：由于操作不慎，可能会造成同时有几个键被按下的现象。

处理串键的技术：

- 1) “两键同时按下”技术是在两个键同时按下时产生保护作用。取最后仍被按下的键是有效的正确按键。
- 2) “n键同时按下”：
 - a、不理睬所有被按下的键，直至只剩下一键按下为止；
 - b、将所有按下的按键信息全部存入内存，再分别处理；
- 3) “n键锁定”技术只处理一个键，通常第一个被按下或最后一个松开的键作为有效按键。

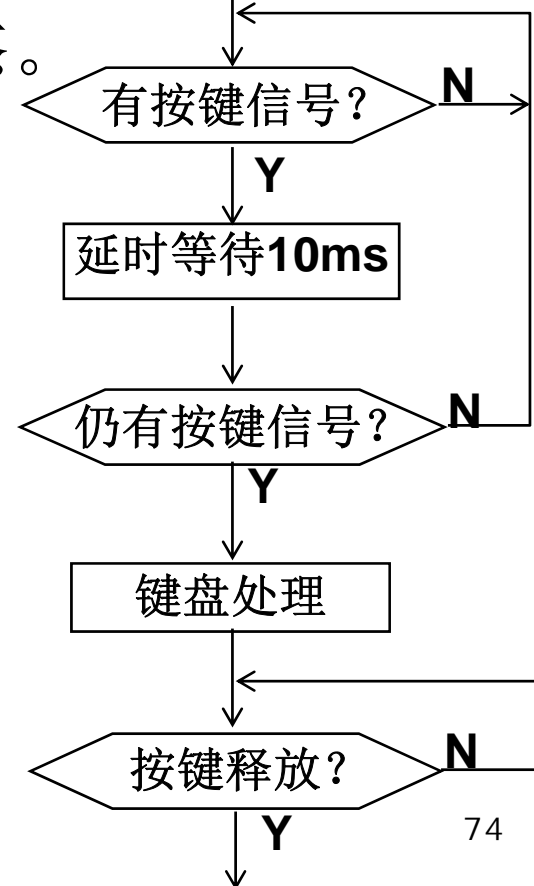
§ 9.4.2 独立式键盘接口设计

■ 独立式键盘接口

特点：每个按键接单片机的一条I/O线，通过对输入线的查询，可以识别每个按键的状态。

结构简单，键盘扫描程序也简单：

- (1) 判断键盘上是否有键闭合；
- (2) 消除键的机械抖动；
- (3) 确定闭合键的物理位置；
- (4) 得到闭合键的编号；
- (5) 确保CPU对键的一次闭合响应

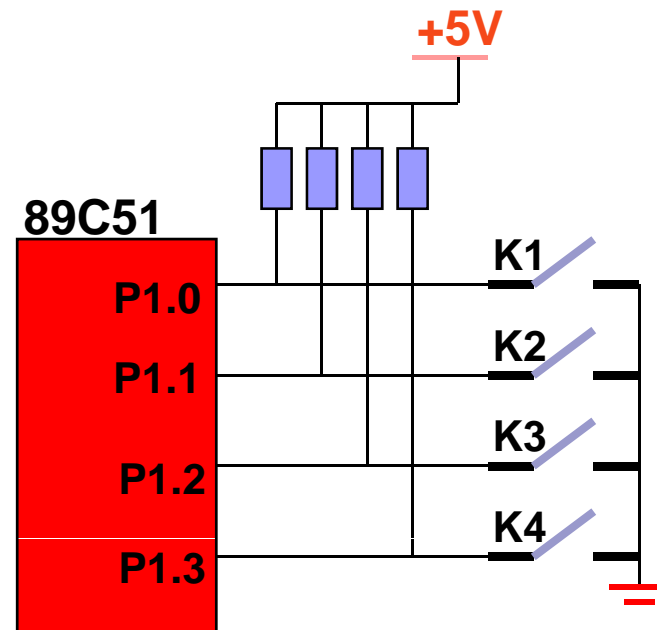


§ 9.4.2 独立式键盘接口设计

独立式键盘的工作原理：当键未被按下时，与此键相连的I/O线获得高电平；当键被按下时，与此键相连的I/O线获得低电平，单片机只要读取I/O口状态，就可以获取按键信息，识别有无键按下和哪个键被按下。

键处理程序如下：

```
MOV P1,#0FFH
UP1: MOV A,P1      ; 读I/O口状态
ANL A,#0FH        ; 屏蔽无用位
CJNE A,#0FH,NEXT1 ; 有闭合键?
SJMP UP1
```



§ 9.4.2 独立式键盘接口设计

```
NEXT1:    LCALL D10ms          ; 延时10ms去抖动
          MOV  A,P1           ; 再读I/O口状态
          ANL  A,#0FH
          CJNE A,#0FH,NEXT2    ; 有闭合键?
          SJMP UP1

NEXT2:    JB   P1.0,NEXT3      ; K1按下?
          LCALL K1             ; K1键处理程序

NEXT3:    JB   P1.1,NEXT4      ; K2按下?
          LCALL K2             ; K2键处理程序

NEXT4:    JB   P1.2,NEXT5      ; K3按下?
          LCALL K3             ; K3键处理程序

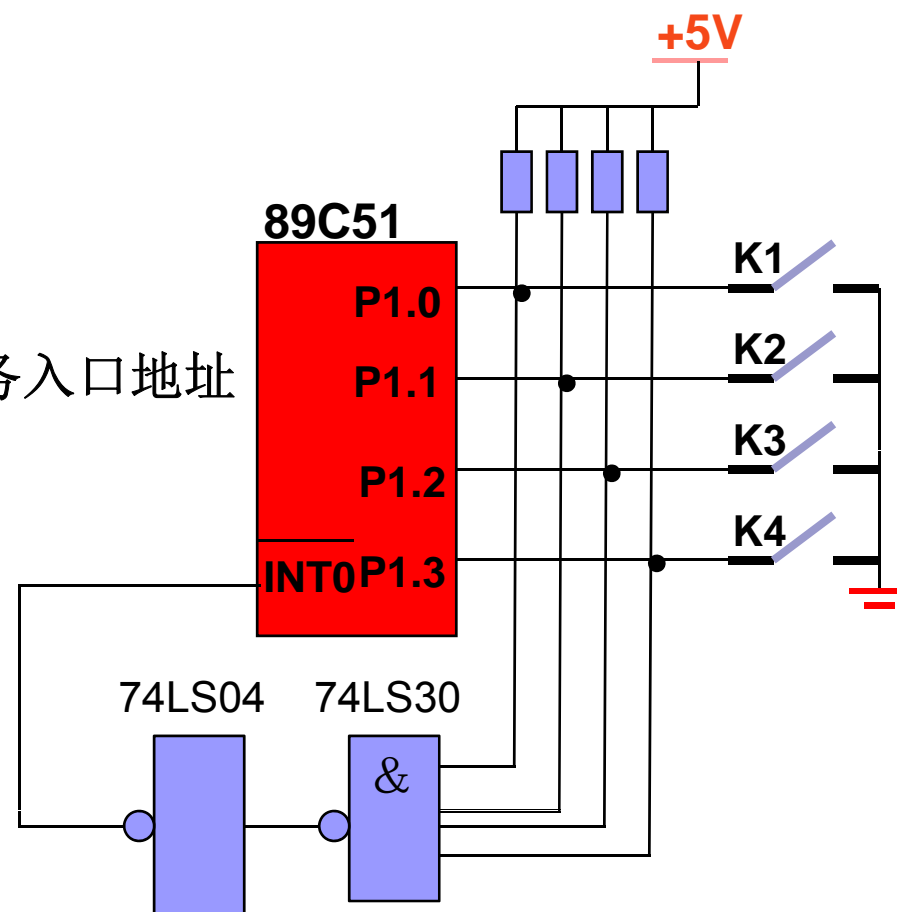
NEXT5:    JB   P1.3,UP1        ; K4按下?
          LCALL K4             ; K4键处理程序
          LJMP UP1
```

§ 9.4.2 独立式键盘接口设计

■ 中断工作方式处理键盘。

主程序如下：

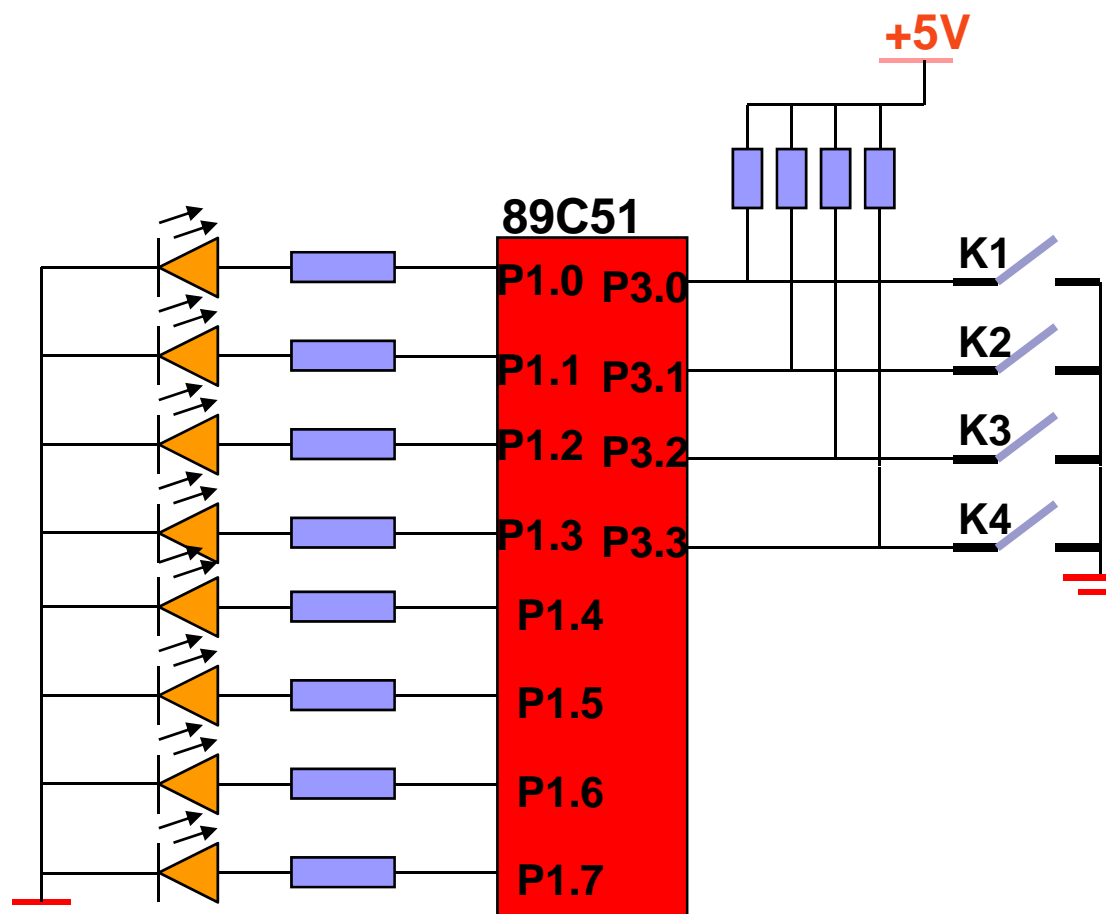
```
ORG    0000H
LJMP   MAIN
ORG    0003H; 外部中断0中断服务入口地址
LJMP   INT  ; 转中断服务
ORG    0100H
MAIN:  SETB  EA    ; 开总中断允许
       SETB  EX0   ; 开INT0中断
       SETB  IT0   ; 下降沿有效
       .....
```



§ 9.4.2 独立式键盘接口设计

```
INT:      LCALL D20ms      ; 延时去抖动
          MOV P1, #0FFH    ; P1口送全1值
          MOV A, P1        ; 读P1口各引脚
          CJNE A, #0FFH, CLOSE ; 验证是否确实有键闭合
          AJMP OUT         ; 无键按下
CLOSE:    JNB ACC.3, KEY 3  ; 查询3号键
          JNB ACC.2, KEY 2  ; 查询2号键
          JNB ACC.1, KEY 1  ; 查询1号键
          JNB ACC.0, KEY 0  ; 查询0号键
OUT:      RETI
KEY 3:    .....          ; 3号键处理程序
KEY 31:   MOV A, P1        ; 再读P1口各引脚
          JNB ACC.3, KEY 31 ; 确认键是否释放
          RETI
KEY 2:    .....          ; 其他键处理程序
          .....
D20ms:    .....          ; 20ms延时子程序
          END
```

例 某单片机控制系统，**P1**口接有**8**发光二极管，**P3.0**、**P3.1**、**P3.2**、**P3.3**接有**4**个开关**K1**、**K2**、**K3**、**K4**，试画出接口电路，并编程使得当**K1**按下时**8**个发光二极管全亮；当**K2**按下时**8**个发光二极管闪亮；当**K3**按下时**8**个发光二极管由左向右点亮；当**K4**按下时**8**个发光二极管全灭。





```
MOV B, #01H
MOV P3, #0FFH
MOV P1, #00H ; 初始全灭
UP1: MOV A, P3
      ANL A, #0FH
      CJNE A, #0FH, NEXT1
      SJMP UP1
NEXT1: LCALL D10ms ; 调用延时子程序
      MOV A, P3
      ANL A, #0FH
      CJNE A, #0FH, NEXT2
      SJMP UP1
NEXT2: JB P3.0, NEXT3
      LCALL K1
NEXT3: JB P3.1, NEXT4
      LCALL K2
NEXT4: JB P3.2, NEXT5
      LCALL K3
NEXT5: JB P3.3, UP1
      LCALL K4
      LJMP UP1
```

子程序

```
K1: MOV P1, #0FFH
      RET
K2: MOV P1, #0FFH
      LCALL D2S
      MOV P1, #00H
      LCALL D2S
      RET
K3: MOV P1, B
      LCALL D2S
      MOV A, B
      RL A
      MOV B, A
      RET
K4: MOV P1, #00H
      RET
```




§ 9.4.3 矩阵式键盘接口设计

- 矩阵式键盘按键识别方法有：
行扫描法和线路反转法。

1、行扫描法

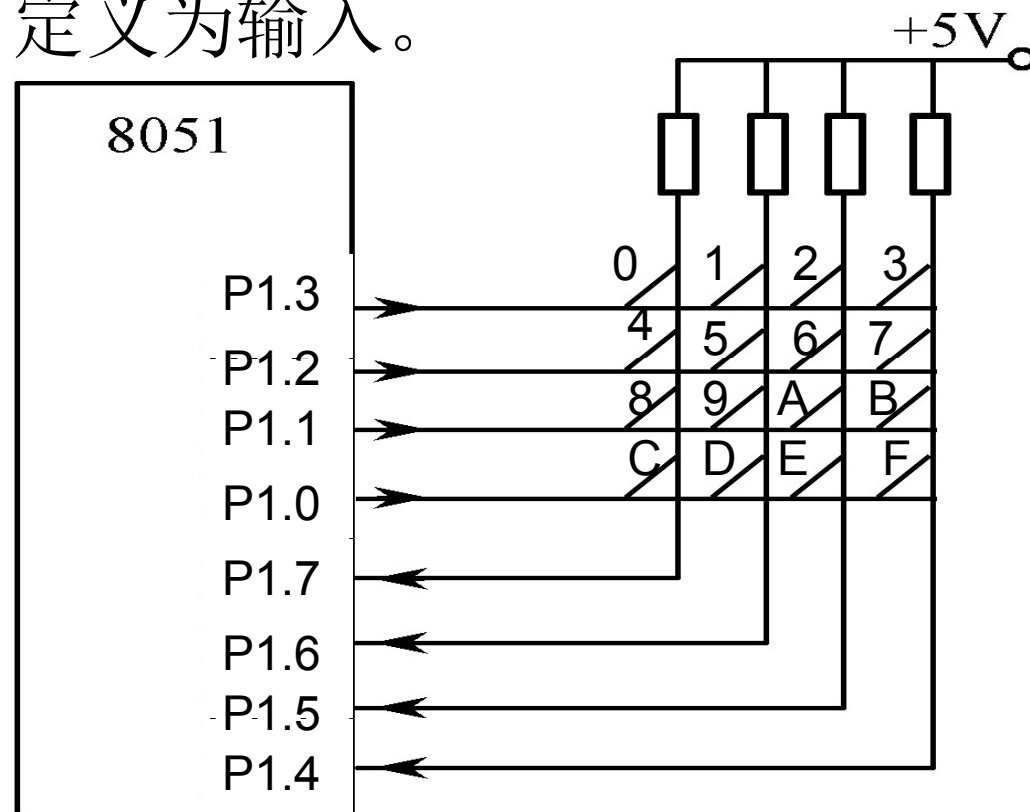
设置行线为输出，列线为输入，当无按键按下时，列输入全为“1”。设计时，将某一行输出为“0”，读取列线值，若其中某一位为“0”，则表明行、列交叉点处的按键被按下，否则无按键按下；继续扫描下一行（将下一行输出为“0”），直至全扫描完为止。

或同理可设置列线为输出，行线为输入。设计时，将某一列输出为“0”。

§ 9.4.3 矩阵式键盘接口设计

例：如图所示为一个 4×4 的矩阵键盘。

P1.3~P1.0作为行线，定义为输出，P1.7~P1.4作为列线，定义为输入。



§ 9.4.3 矩阵式键盘接口设计

行线和列线的状态编码可以唯一地表征某个键被按下，因此可以把二者组合成一个特征码，如当按键2按下时，行线——0111；列线——1101。特征码：1101 0111即D7H。可写出按键的特征码表。程序如下：

```
SCAN_KEYB:  MOV  P1,#0F0H    ;置P1.7~P1.4输入，P1.3~P1.0输出
              MOV  A,P1      ;读P1.7~P1.4引脚状态
              ANL   A,#0F0H
              XRL   A,#0F0H
              JZ     NO_KEY   ;判断有无按键按下
              ACALL D20MS     ;有按键按下延时消抖
              MOV  A,P1      ;重读键盘
              ANL   A,#0F0H
              XRL   A,#0F0H
              JZ     NO_KEY   ;再判有无键按下
```



```
MOV R2,#11110111B ;行扫描初始值，从
SCAN: MOV A,R2
      MOV P1,A
      MOV A,P1
      ANL A,#11110000B
      MOV R3,A ;取列线P1.7~P1.4引脚状态
      CJNE A,#0F0H,KEY_PRSD ;有键按下
      MOV A,R2
      RR A ;产生下次的行线输出
      MOV R2,A
      XRL A,#01111111B
      JNZ SCAN ;4行已扫描完？
NO_KEY: MOV R5,#0F0H ;无按键按下
        RET
KEY_PRSD: MOV A,R2 ;取行扫描值
          ANL A,#00001111B ;计算行特征码
          ORL A,R3 ;计算按键的特征码
```



```
MOV R4,A
MOV R5,#00H           ;设置按键键值初值
MOV DPTR,#KEY_TAB     ;特征码表首地址
CAL_VAL: MOV A,R5
MOVC A,@A+DPTR
XRL A,R4
JZ FIXED              ;键值求出存在（R5）
INC R5
SJMP CAL_VAL
FIXED: MOV A,P1        ;判断键是否释放
ANL A,#0F0H
XRL A,#0F0H
JNZ FIXED
ACALL DL20MS
MOV A,P1
ANL A,#0F0H
```



```
XRL  A,#0F0H
```

```
JNZ  FIXED
```

```
RET
```

```
KEY_TAB:  DB  77H,0B7H,0D7H,0E7H,7BH,0BBH,0DBH,0EBH  
           DB  7DH,0BDH,0DDH,0EDH,7EH,0BEH,0DEH,0EEH
```



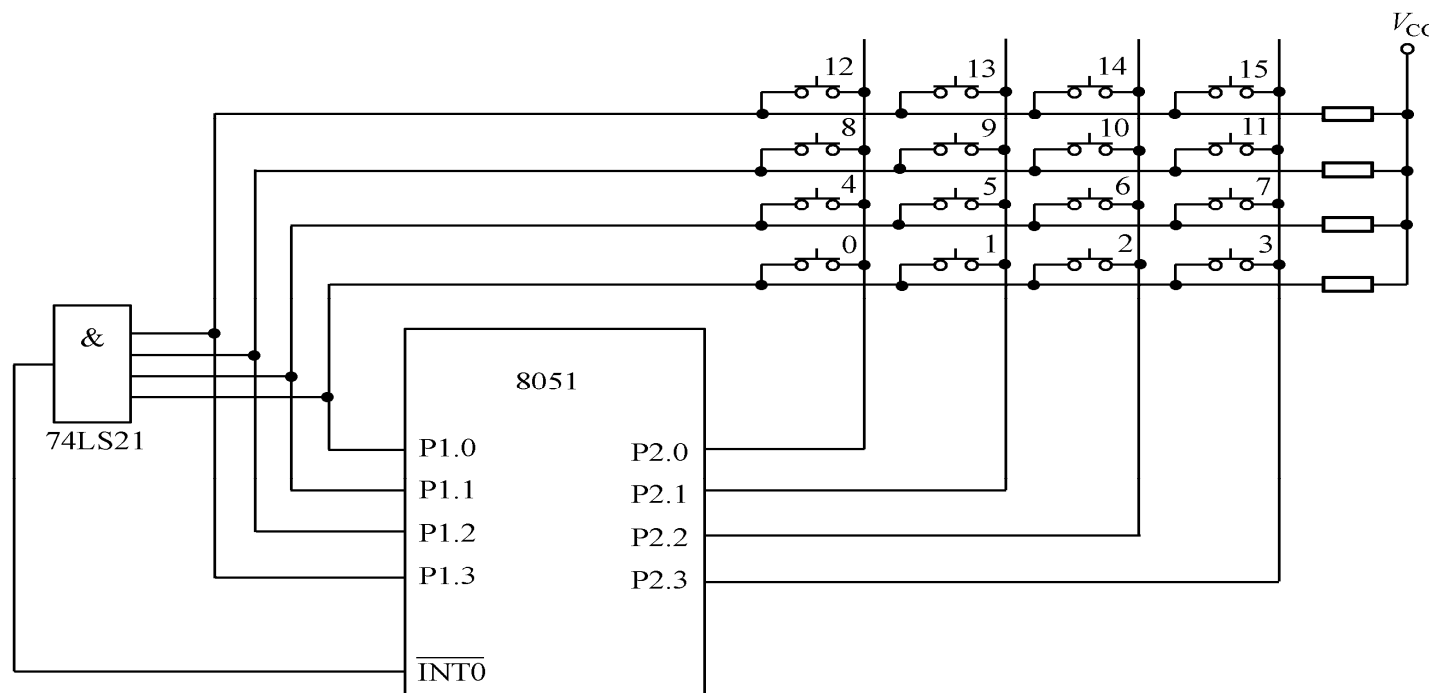
§ 9.4.3 矩阵式键盘接口设计

- 线路反转法需要两个双向I/O口分别接行、列线。
步骤如下：
 - (1) 由行线输出全“0”，读入列线，判有无键按下（若有一个列为“0”则表明有键按下）。
 - (2) 若有键按下,再将读入的列值从列线输出,读取行线的值。
 - (3) 第一步读入的列值与第二步读入的行值运算，
从而得到代表此键的唯一的特征值。

优点：判键速度快，两次即可。

§ 9.4.3 矩阵式键盘接口设计

4×4 矩阵键盘的线路连接如图所示。**P2**口的低4位作为行线(初始输出全“0”)。**P1**口的低4位作为列线,通过**74LS21**进行逻辑相与后作为外部中断源输入。当有键按下时就将引起中断。中断服务程序要对所按的键进行判别。





§ 9.4.3 矩阵式键盘接口设计

(1) 查询闭合键的位置子程序**KEYR**

KEYR子程序用以确定每组线中哪一位为0，是否有多个0。在调用前，应将读某组线的数据存入累加器**A**中。**KEYR**子程序返回时，某组线中0的位置（0~3）保存在**R3**中。按键闭合引起中断后，执行中断服务程序。



```
KEYR:    CJNE    A, #0FEH, TESTP11 ; 测试P1.0
          MOV     R3, #0           ; P1.0=0, 说明被按键的输入线为P1.0
          LJMP    FINISH           ; 返回
TESTP11: CJNE    A, #0FDH, TESTP12 ; 测试P1.1
          MOV     R3, #1
          LJMP    FINISH
TESTP12: CJNE    A, #0FBH, TESTP13 ; 测试P1.2
          MOV     R3, #2
          LJMP    FINISH
TESTP13: CJNE    A, #0F7H, FINISH   ; 测试P1.3
          MOV     R3, #3
FINISH:   RET
```

(2) 中断服务程序

中断服务程序开始部分应利用软件延时消除键抖动，然后再对所按的键做出处理。

中断服务程序如下：

```
ORG 1000H
INT11: LCALL DELAY          ; 延时去抖动
      MOV  A, P1            ; 读列线
      ANL  A, #0FH          ; 判断是否有键闭合
      CJNE A, #0FH, TEST    ; 有键闭合，转判断按键程序
      RETI                  ; 无键闭合，返回
TEST:  MOV  B, A            ; 暂存
      LCALL KEYR            ; 调用读取子程序
      MOV  40H, R3          ; 暂存在40H单元
      MOV  P2, #0FFH        ; 行线写1，转输入模式
      MOV  P1, B            ; 列线输出数据
      MOV  A, P2            ; 读行线
      LCALL KEYR            ; 调用读取子程序
      XCH  A, R3
      SWAP A
      ORL  40H, A           ; 得按键特征值
      RETI
```

§ 9.4.3 矩阵式键盘接口设计

中断程序结束后，键的特征值存放在40H单元中。此键的行线号位于40H单元的高4位，其列线号位于低4位。此后，根据40H单元的内容去查表，得到相应键的代码，可进行显示或其他处理。

(3) 去抖动的延时子程序DELAY

利用CPU的空闲方式，通过定时器T1实现延时，T1必须预先置初值，以得到需要的延迟时间。设晶振频率为12MHz，欲延时20ms，定时时间为：

$$(2^{16} - T_C) \times 12/12 = 20 \times 10^3 \mu s, \text{ 初值 } T_C = 45536 = B1E0H.$$

§ 9.4.3 矩阵式键盘接口设计

```
DELAY: MOV    TMOD, #11H    ; 方式1定时
        MOV    TL1, #0E0H   ; 定时器1定时初值
        MOV    TH1, #0B1H
        SETB   EA           ; 开中断
        SETB   ET1          ; 开定时器1中断
        SETB   PT1          ; 定时器1为高级中断（因被键盘中断调用）
        SETB   TR1          ; 启动定时器
        ORL    PCON, #1      ; 启动空闲方式，实际CPU在此处等待
        CLR    TR1          ; 以下四条指令只有在延时后，定时器被唤醒，才能执行
        CLR    PT1
        CLR    ET1
        RET
        END
```



§ 9.5 LED显示接口与设计

- § 9.5.1 LED显示原理
- § 9.5.2 七段LED显示及接口



§ 9.5.1 LED显示原理

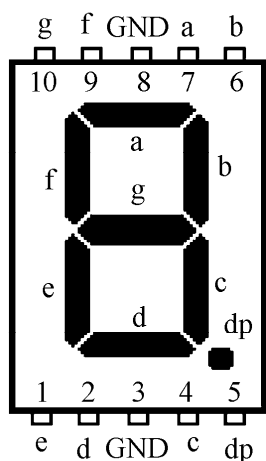
- 发光二极管简称**LED**，是微机系统最常使用的显示器。**LED**显示器有单个、七段和点阵式等几种类型。

1、七段**LED**显示器

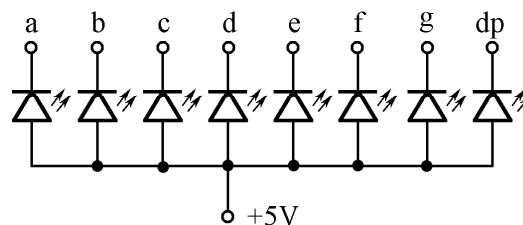
七段**LED**数码管显示器由8只LED发光管组成段码，能够显示十进制或十六进制数字及某些简单字符。但控制简单，使用方便，在单片机系统中应用较多。

为了显示某个数或字符，即要点亮相应的段，要向LED送一定的数值，这种数值称为段码。

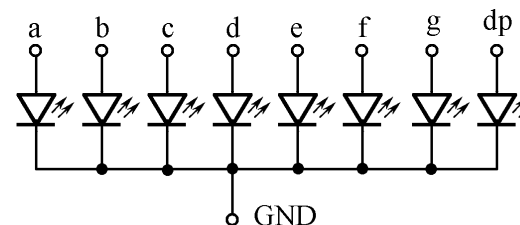
§ 9.5.1 LED显示原理



(a) 外形



(b) 共阳极



(c) 共阴极

上图中的a~g七个笔划（段）及小数点dp均为发光二极管。数码管显示器根据公共端的连接方式，可以分为共阴极数码管（将所有发光二极管的阴极连在一起）和共阳极数码管（将所有发光二极管的阳极连在一起）。

§ 9.5.1 LED显示原理

七段LED字形码如下表所示：

显示字符 ₁₆	共阳极字符 ₁₆	共阴极字符 ₁₆	显示字符 ₁₆	共阳极字符 ₁₆	共阴极字符 ₁₆
0 ₁₆	C0H ₁₆	3FH ₁₆	A ₁₆	88H ₁₆	77H ₁₆
1 ₁₆	F9H ₁₆	06H ₁₆	b ₁₆	83H ₁₆	7CH ₁₆
2 ₁₆	A4H ₁₆	5BH ₁₆	C ₁₆	C6H ₁₆	39H ₁₆
3 ₁₆	B0H ₁₆	4FH ₁₆	d ₁₆	A1H ₁₆	5EH ₁₆
4 ₁₆	99H ₁₆	66H ₁₆	E ₁₆	86H ₁₆	79H ₁₆
5 ₁₆	92H ₁₆	6DH ₁₆	F ₁₆	8EH ₁₆	71H ₁₆
6 ₁₆	82H ₁₆	7DH ₁₆	P ₁₆	8CH ₁₆	73H ₁₆
7 ₁₆	F8H ₁₆	07H ₁₆	H ₁₆	89H ₁₆	76H ₁₆
8 ₁₆	80H ₁₆	7FH ₁₆	L ₁₆	C7H ₁₆	38H ₁₆
9 ₁₆	90H ₁₆	6FH ₁₆	“灭” ₁₆	FFH ₁₆	00H ₁₆



§ 9.5.2 七段LED显示及接口

■ LED数码管的工作方式

LED数码管有静态显示和动态显示两种形式。

(1) 静态显示方式

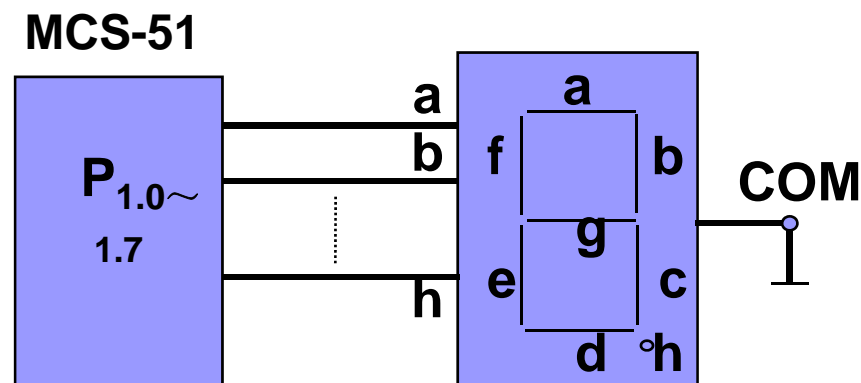
静态显示方式的各数码管在显示过程中持续得到送显信号，与各数码管接口的**I/O**口线是专用的。其特点是显示稳定，无闪烁，用元器件多，占**I/O**线多，无须扫描。系统运行过程中，在需要更新显示内容时，**CPU**才去执行显示更新子程序，节省**CPU**时间，提高**CPU**的工作效率，编程简单。

§ 9.5.2 七段LED显示及接口

1)设置显示缓冲区，存放待显示数据。

2)显示译码：程序存储器中建立字形码常数表，查表得出对应数据的字形码。

3)输出显示：输出字形码到显示端口。



MOV DPTR, #WTAB ; 指向字形码表首地址

MOV A, @R0 ; 取显示缓冲区中数据

MOVC A, @A+DPTR ; 查表显示译码

MOV P1, A ; 输出显示

...

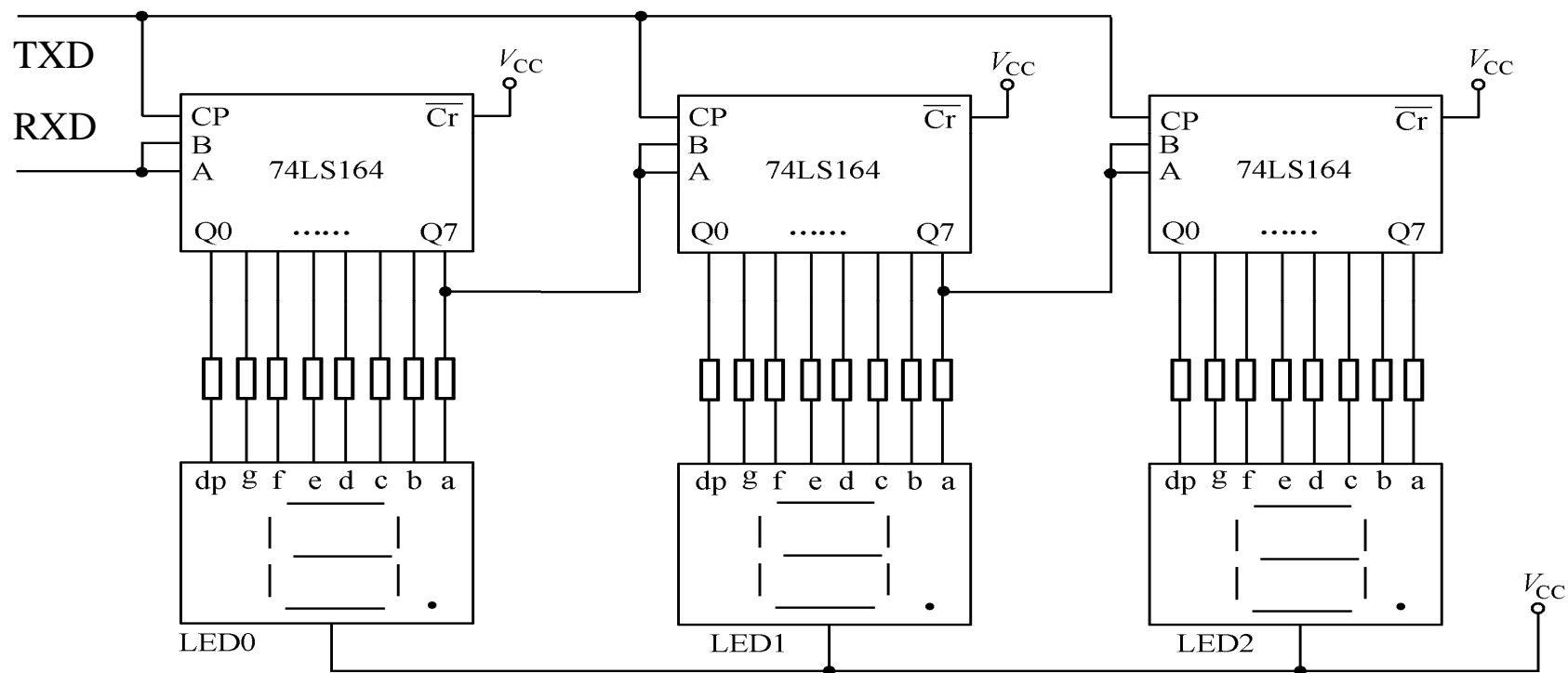
WTAB: DB 3FH, 06H, 5BH ; 字形码表

...



§ 9.5.2 七段LED显示及接口

- **[例题]** 利用在8051串行口扩展多片串行输入并行输出的移位寄存器74LS164作为静态显示器接口的方法，设计3位静态显示器接口，并写出显示更新子程序，实现将7FH~7DH 3个单元的数值分别显示在3位LED2~LED0上。



■ **解：**接口电路如图所示。3个共阳极数码管的公共端均接Vcc，段码通过串行口，采用串—并转换原理，分别送出3个数码管的段码（先送出的段码字节在LED2数码管上显示），图中的电阻值100~500Ω。



程序如下：

```
                ORG    1000H
DISPSE: MOV     R5, #03H    ; 显示3个字符
                MOV     R1, #7FH    ; 7FH~7DH存放要显示的数据
DL0:  MOV     A, @R1        ; 取出要显示的数据
                MOV     DPTR, #STAB ; 指向段数据表
                MOVC    A, @A+DPTR  ; 查表取字形数据
                MOV     SBUF, A      ; 送出数据，进行显示
                JNB     TI, $        ; 输出完否？
                CLR     TI          ; 输出完，清中断标志
                DEC     R1          ; 再取下一个数据
                DJNZ    R5, DL0      ; 循环3次
                RET                ; 返回
STAB:  DB      0C0H, 0F9H, 0A4H, 0B0H; 段数据表（共阳极）
                DB      .....
                .....
END
```



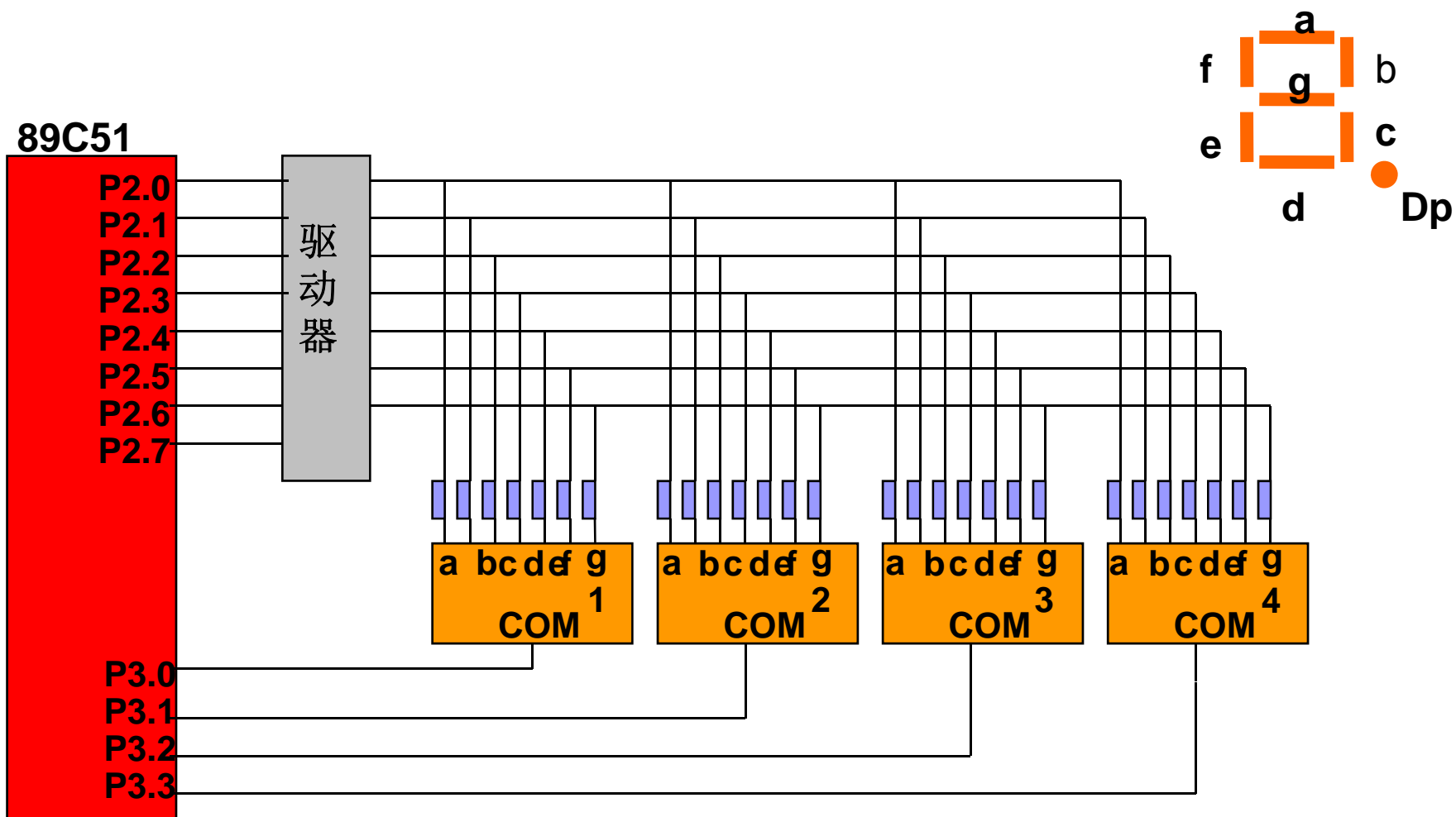
§ 9.5.2 七段LED显示及接口

(2) 动态显示方式

动态显示方式是指一位一位地轮流点亮每位显示器，与各数码管接口的**I/O**口线是共用的。其特点是有闪烁，用元器件少，占**I/O**线少，必须扫描，花费**CPU**时间，编程复杂。



例：编程在4个七段LED数码管上显示30H，31H，32H，33H单元中的内容。

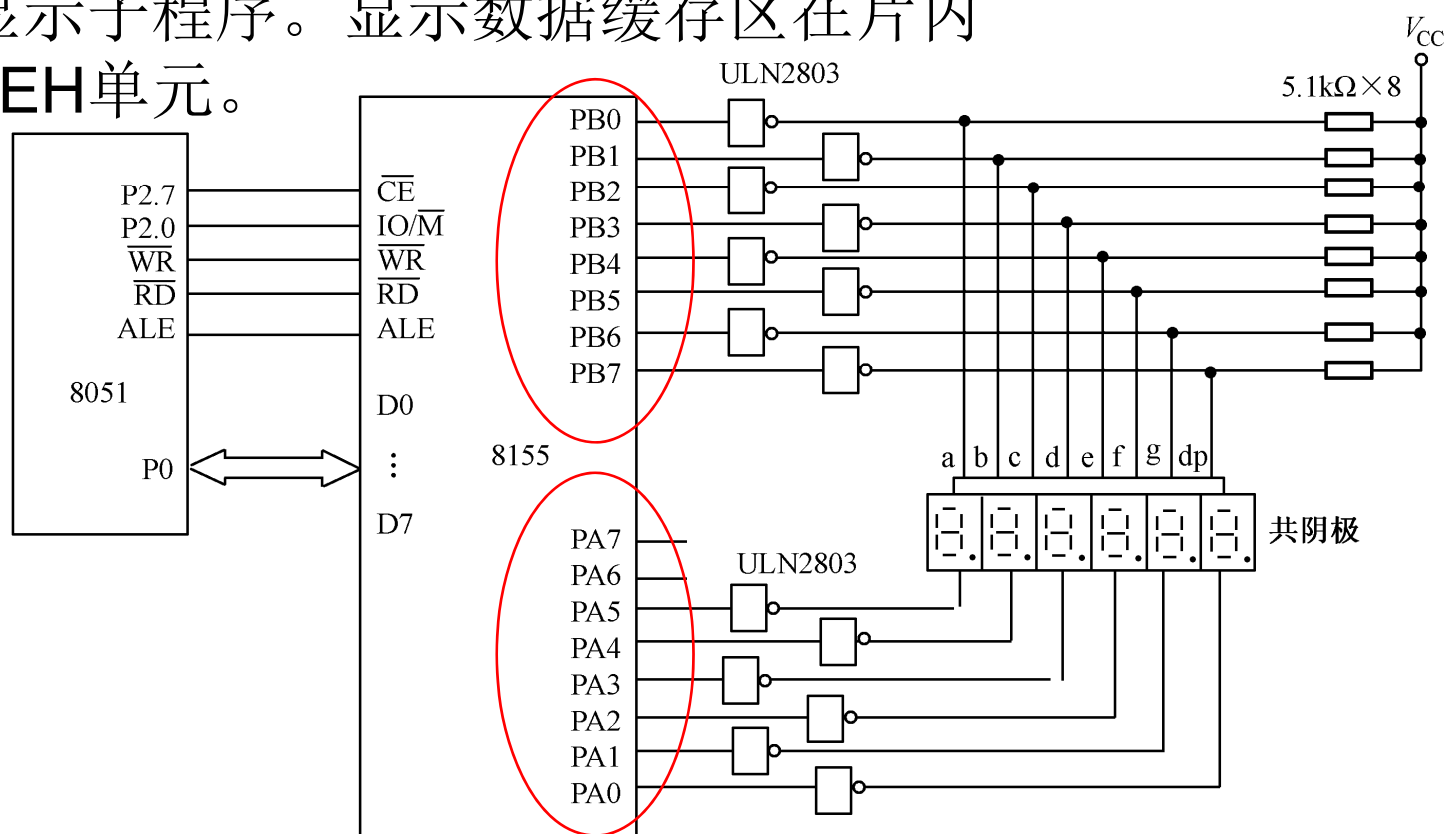




```
START: MOV R0,#30H
        MOV R7,#4
        MOV R2,#0FEH
        MOV DPTR,#TAB
UP:     MOV A,@R0
        MOVC A,@A+DPTR
        MOV P2,A
        MOV P3,R2
        LCALL D2MS
        INC R0
        MOV A,R2
        RL A
        MOV R2,A
        DJNZ R7,UP
        SJMP START
TAB : DB 3FH,06H,5BH,4FH
      DB 66H,6DH,7DH,07H
      DB 7FH,6FH
```

§ 9.5.2 七段LED显示及接口

[例题]设计6位共阴极显示器与8155的接口电路,并写出与之对应的动态扫描显示子程序。显示数据缓存区在片内RAM79H~7EH单元。



设计8155的PA作为扫描口，PB作为段码输出口，都工作在基本输出方式下，PA口的端口地址为7F01H，PB口的端口地址为7F02H。进行扫描时，PA的低6位依次置1，依次选中了从左至右的显示器。使用ULN2803作为段码输出驱动（反相驱动），所以共阴极数码管在段数据表中的字形码应与共阳极数码管的字形码相同。

动态扫描子程序如下:

```
                ORG    1000H
DSP8155:  MOV    DPTR, #7F00H    ;指向8155命令寄存器
                MOV    A, #00000011B    ;设定PA口、PB口输出方式
                MOVX   @DPTR, A    ;输出命令字
DISP1:    MOV     R0, #7EH    ;指向缓冲区末地址
                MOV     A, #20H    ;扫描字, PA5为1, 从左至右扫描
LOOP:     MOV    R2, A
                MOV    DPTR, #7F01H    ;指向8155的PA
                MOVX   @DPTR, A    ;输出位选码
                MOV    A, @R0
                MOV    DPTR, #PTRN    ;指向段数据表首地址
                MOVC   A, @A+DPTR
                MOV    DPTR, #7F02H    ;指向8155的PB
                MOVX   @DPTR, A    ;输出段数据
                CALL   D1MS    ;延时1ms
                DEC    R0
                MOV    A, R2    ;读回扫描
```



```

                                CLR  C                ;清进位标志
                                RRC  A                ;扫描字右移
                                JC   PASS
                                AJMP LOOP
PASS:                            RET

D1MS:                            MOV  R7, #02H        ;延时1ms子程序
DMS:                             MOV  R6, #0FFH
                                DJNZ R6, $
                                DJNZ R7, DMS
                                RET

PTRN:                            DB      0C0H, 0F9H, 0A4H, 0B0H, 99H    ;段数据表
                                DB      .....
                                DB      .....
                                .....
                                END
```